(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 726 532 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.08.1996 Bulletin 1996/33

(51) Int Cl.[6]: **G06F 15/80**, G06F 15/16

(21) Application number: **96480018.9**

(22) Date of filing: **06.02.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **10.02.1995 US 386384**

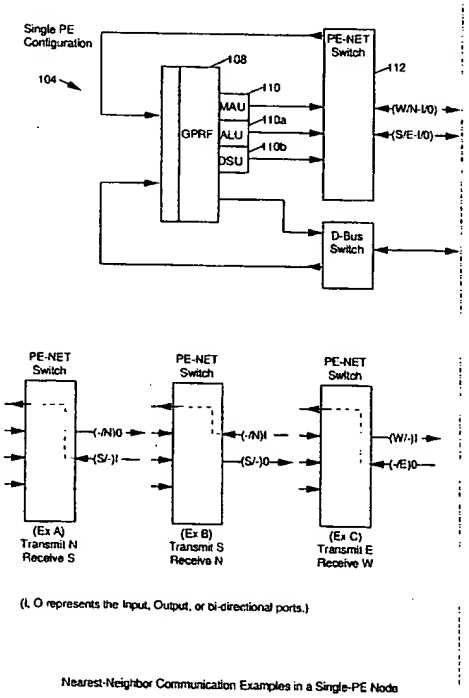
(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

(72) Inventors:
- **Pechanek, Gerald G.**
  Cary, NC 27511 (US)
- **Vassiliadis, Stamatis**
  NL-2726 Zoetermeer (NL)
- **Glossner, Clair J.**
  Durham, NC 27707 (US)
- **Larsen, Larry D.**
  Raleigh, NC 27609 (US)

(74) Representative: **Lattard, Nicole**
Compagnie IBM France
Département de Propriété Intellectuelle
06610 La Gaude (FR)

(54) Array processor communication architecture with broadcast instuctions

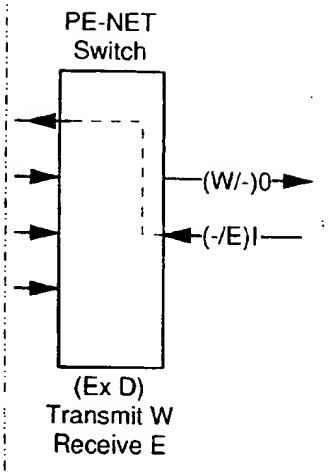
(57) A plurality of processor elements (PEs) are connected in a cluster by a common instruction bus to an instruction memory. Each PE has data buses connected to at least its four nearest PE neighbors, referred to as its North, South, East and West PE neighbors. Each PE also has a general purpose register file containing several operand registers. A common instruction is broadcast from the instruction memory over the instruction bus to each PE in the cluster. The instruction includes an opcode value that controls the arithmetic or logical operation performed by an execution unit in the PE on an operand from one of the operand registers in the register file. A switch is included in each PE to interconnect it with a first PE neighbor as the destination to which the result from the execution unit is sent. The broadcast instruction includes a destination field that controls the switch in the PE, to dynamically select the destination neighbor PE to which the result is sent. Further, the broadcast instruction includes a target field that controls the switch in the PE, to dynamically select the operand register in the register file of the PE, to which another result received from another neighbor PE in the cluster is stored. In this manner, the instruction broadcast to all the PEs in the cluster, dynamically controls the communication of operands and results between the PEs in the cluster, in a single instruction, multiple data processor array.



Nearest-Neighbor Communication Examples in a Single-PE Node

FIG 6A

EP 0 726 532 A2



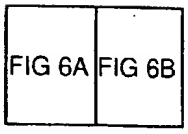


FIG 6

FIG 6B

Description

FIELD OF THE INVENTION

The invention disclosed broadly relates to data processing systems and method and more particularly relates to improvements in array processor architectures.

OBJECTS OF THE INVENTION

It is therefore an object of the invention to provide an improved instruction driven programmable parallel processing system.

It is another object of the invention to provide an improved parallel processing system that reduces the inherent latency in communicating between procesors.

It is still a further object of the invention to provide an improved parallel processing system which has improved performance characteristics.

SUMMARY OF THE INVENTION

These and other objects, features and advantages are accomplished by the invention. A plurality of processor elements (PEs) are connected in a cluster by a common instruction bus to an instruction memory. Each PE has data buses connected to at least its four nearest PE neighbors, referred to as its North, South, East and West PE neighbors. Each PE also has a general purpose register file containing several operand registers. A common instruction is broadcast from the instruction memory over the instruction bus to each PE in the cluster. The instruction includes an opcode value that controls the arithmetic or logical operation performed by an execution unit in the PE on an operand from one of the operand registers in the register file. A switch is included in each PE to interconnect it with a first PE neighbor as the destination to which the result from the execution unit is sent. In accordance with the invention, the broadcast instruction includes a destination field that controls the switch in the PE, to dynamically select the destination neighbor PE to which the result is sent. Further in acordance with the invention, the broadcast instruction includes a target field that controls the switch in the PE, to dynamically select the operand register in the register file of the PE, to which another result received from another neighbor PE in the cluster is stored. In this manner, the instruction broadcast to all the PEs in the cluster, dynamically controls the communication of operands and results between the PEs in the cluster, in a single instruction, multiple data processor array.

DESCRIPTION OF THE FIGURES

Fig. 1 is a high level array machine organization diagram for multiple control units.

Fig. 2 is an example instruction format for communications, in accordance with the invention.

Fig. 3 is a second example instruction format for communications, in accordance with the invention.

Fig. 4 is a single processor element (diagonal) node flow with connection interfaces.

Fig. 5 is a dual processor element node flow with connection interfaces.

Fig. 6 is a nearest-neighbor communication example in a single processor element node.

Fig. 7 is a nearest-neighbor communication example in a dual processor element node.

Fig. 8 is a logical and folded mesh representation of adjacent processor element column communications.

Fig. 9 is a logical and folded mesh representation of adjacent processor element row communications.

Fig. 10 is a flow organization of a processing element, depicting the flow of an individual processing element, showing all its arithmetic facilities and the points to find for connection to switch logic and paired processor elements.

Fig. 11 is a general form diagram of the data select unit.

Fig. 12 is an example of a data select unit in use, wherein byte B of the source is placed in the low order 8 bits of the destination register and all the remaining destination bits are forced to be the same as the sign of the byte, as performed by the data selector logic.

Fig. 13 is a flow chart showing folded array fast odd/even symmetric 1-D DCT.

Fig. 14 illustrates the execution of a butterfly surrogate.

Fig. 15 illustrates the first execution of multiply add and DSU send surrogate.

Fig. 16 illustrates the second execution of multiply add and DSU send surrogate.

Fig. 17 illustrates the third execution of multiply add and DSU send surrogate.

Fig. 18 illustrate the fourth execution of multiply add and DSU send surrogate.

Fig. 19 illustrates the execution of the butterfly with a clustered processor element destination surrogate.

I-Bus Switch

0 = IA-Bus Port connects to the Top PE, IB-Bus Port connects to the Bottom PE
1 = IA-Bus Port connects to the Bottom PE, IB-Bus Port connects to the Top PE D-Bus Switch
0 = DA-Bus Port connects to the Top PE, DB-Bus Port connects to the Bottom PE
1 = DA-Bus Port connects to the Bottom PE, DB-Bus Port connects to the Top PE Nearest Neighbor/Adjacent-PE Communications Mode
0 = Nearest Neighbor Communications Enabled
1 = Adjacent-PE Communications Enabled Ring/Array Mode
00 = Row Rings: N and S Ports Disabled, E and W Ports Enabled
01 = Column Rings: E and W Ports Disabled, N and S Ports Enabled
10 = Reserved
11 = Array Mode A Load Offset Register instruction provides byte loading of the PE's offset register used in Case Surrogate instructions.

The PE flow diagrams depicted in Fig. 4 and Fig. 5 show switches which are controlled from instructions being executed. Three types of switches are indicated in the nodes: the PE-NET Switch, Data-Bus (D-Bus) Switch, and Instruction-Bus (I-Bus) switch (only in the dual-PE node). The PE-NET switches are controlled by instructions executing in the PEs; the I-Bus and D-Bus switches are controlled by the PE-mode control register. The PEs exchange data in different ways among nodes by controlling the PE-NET switches.

The Nearest-Neighbor (PE-NET) transmitting/receiving facilities for dual-PE nodes include four bi-directional or four input and four output driver/receiver ports. For the separate input/output port case, there is an input and output pair assigned to each nearest neighbor direction. In general, when one of these ports is assigned to transmit data to a nearest-neighbor PE, another port will also be directed to simultaneously receive data from a different PE. In the dual-PE node, Controls are provided to assure that only two of the driver/receiver ports are simultaneously transmitting data and two others are simultaneously receiving data. There are four specific cases to consider: they are Transmit East Receive West, Transmit North Receive South, Transmit South Receive North, and Transmit West Receive East. The diagonal PEs (Fig. 4) share the West/North ports and the South/East ports and thus require only two nearest-neighbor type I/O ports per diagonal PE. Note that in the dual-PE nodes, the receiving/transmitting mechanisms consist of four I/O ports.

Fig. 6 and Fig. 7 depict the single-PE and dual-PE nodes in a less detailed form than Fig. 4 and Fig. 5 and show examples of how the nearest-neighbor ports are used in supporting four possible transmission modes. The communication modes supported cause data to be transmitted in the four cardinal directions and simultaneously received from the opposite direction. For example, transmit North and receive from the South.

Fig. 8 and Fig. 9 depict, for a four neighborhood array, the logical representation (i.e. on the unfolded mesh) and the folded mesh representation of the four communications modes possible:

Col-Ring 0 -- Col-Ring 1 & Col-Ring 2 -- Col-Ring 3 Figs. 8A and 8B.
Col-Ring 0 -- Col-Ring 3 & Col-Ring 1 -- Col-Ring 2 Fig. 8C and 8D.
Row-Ring 0 -- Row-Ring 3 & Row-Ring 1 -- Row-Ring 2 Fig. 9C and 9D
Row-Ring 0 -- Row-Ring 1 & Row-Ring 2 -- Row-Ring 3 Fig. 9D and 9B

The folded-array nodes each contain one or two Processing Elements as shown in Fig. 4 and Fig. 5. The PEs, which are all identical, each contain two general types of arithmetic units &dash. an ALU and a fixed point/floating point Multiply/Add Unit (MAU) &dash. a Data Selection Unit (DSU), a Local Load Data Address Generator, a Local Store Data Address Generator, and a set of 32 GPRs which serves to hold operands and working results for operations performed in the node. The register set is referred to as the General Purpose Register File, or GPRF for short. A view of the PE data flow organization, showing its individual processing elements and GPR file, is shown in Fig. 10.

Three classes of Multiply/Add Unit MAU instructions are architectured, one for 16x16 single 32-bit fixed point results, a second for 32x32/dual-16x16 double 32-bit fixed point results, and a third for Single Precision Real Single Precision Real floating point multiply-add results. An array processor can be designed to support any one of these options and operate as subsets of the full architecture. With a 64-bit result (or double 32-bit results) the low 32-bit half uses bus Q. For an implementation with a 16x16 MAU only, the Q bus is not used. For a 32-bit Mfast processor the 32x32 MAU instructions are only able to write their results back to a local GPRF. The DEST destination field is not used in 32x32 MAU instructions and the second 32-bit result is written back to the target register specified plus one. Use of the clustered communications and nearest neighbor interface for the 32x32 MAU instructions is reserved for future machines with 64-bit Nearest Neighbor ports. Note that the Processing Element flow does not include any of the switch logic required for communication among array nodes; that logic is unique to the nodes themselves. The

idea, here, is that the logic for the PE can be designed as a macro and used repeatedly to build the array. It is intended for the PE to be designed with 6 unconnected GPRF input ports, and allow these inputs to be connected in the manner appropriate for the particular node in which the PE is included. Fig. 10 indicates typical direct connections (shown in dashed lines) from the ALU, MAU, and DSU. A more complete picture of these direct connections is shown in the Single-PE node flow diagram (Fig. 4).

The GPRF input ports may also be multiplexed as shown in the Dual-PE node flow diagram (Fig. 5).

The Data Select Unit (DSU) is used in register-to-register moves and data shift operations. In the specific situation when the move destination is also the source for another move (registers are co-destinations between paired-PEs), a SWAP function can be achieved. The general form of the DSU is shown in Fig. 11 :

The logic in the Data Selector is used to modify the data passing from source to destination for those instructions which require it. For example, when a byte is selected from the source register and then loaded into the destination in sign-extended form, the Data Selector will perform the byte-alignment and sign-extension functions. A simple example of this kind of operation is shown in Fig. 12.

Three types of data select/move operations are provided by the Data Selector: word move, halfword move, and byte move. Within these types of moves, certain variations are supported:

Word move
Halfword move

Any source halfword to any destination halfword
Any source halfword to low half of word and:

High halfword forced to all zeros
High halfword forced to all ones
High halfword forced to low halfword sign value

Byte move

Any source byte to any destination byte
Any source byte to low destination byte and:

High bytes forced to all zeros
High bytes forced to all ones
High bytes forced to low byte sign value

High or low source byte pairs (b0 and bl, b2 and b3) to destination bytes bl and b3, and:

High bytes forced to all zeros
High bytes forced to all ones
High bytes forced to low byte sign value

When a PE execution unit performs an operations, the resulting output (P, U, and/or S) is sent to a destination register which may be in the same PE as the execution unit (a local register), in a paired-PE (in a dual-PE node), in a nearest-neighbor (NN) PE, or in an adjacent-PE. In all cases, the destination register (Rm, Rt, and/or Rs) is specified in conjunction with the instruction's Destination (DEST) field. Table 1 and Table 2 list the destination (DEST) options presently defined. This field may be modified for specific instructions. For surrogate instructions only one execution unit is specified to use the nearest neighbor interface.

A folded array 2 dimensional (2D) discrete cosine transform is prented next.

The signal flow graph for the symmetric discrete cosine transform (DCT) is shown in Fig. 13. Note that the outputs are scaled by the 2C(u)/N where C(u)=1/sqrt 2 for u=0 and C(u)=1 otherwise. Note that c#x=cos(# pi/16) and that 1/(4 sqrt 2) = 1/4 c4x.

For the 2-D DCT a 1-D DCT on the columns followed by a 1-D DCT on the Rows produces the 2-D DCT result. A multiplication accumulate and register transfer via the nearest neighbor ports procedure is used. Since the butterfly results are 16-bits and the nearest neighbor ports are 32-bits both odd and even butterfly values can be sent between PEs each cycle. With dual 16x16 multipliers in each PE both odd and even parts of the 1-D DCT for 4-columns can be calculated in the same four cycles. The column 1-D DCT equations shown in Table 3. The lower case "z" in the following lists represents the column # being processed. Fig. 14 illustrates the execution of the butterfly surrogate. Fig.

15 illustrates the first execution of the multiply add and DSU send surrogate. Fig. 16 illustrates the second execution of the multiply add and DSU send surrogate. Fig. 17 illustrates the third execution of the multiply add and DSU send surrogate. Fig. 18 illustrates the fourth execution of the multiply add and DSU send surrogate. Fig. 19 illustrates the execution of the butterfly with the clustered processor element destination surrogate.

Another 4-cycles finishes all 8-column 1-D DCTs. The scaling of the output by 1/4 is not done at this point but rather the combined scaling of 1/16 for the 2-D DCT is included in the calculation of the quantization table values. Consequently, the procedure continues with doing the 1-D DCT on the row values. First a half-word butterfly surrogate instruction creates all the row butterfly values placing the values in the pattern shown in Fig. 19 where "z", of Az through Hz, represents the row # now instead of the column #. Note that the half-word butterfly instructions send their results to the paired-PE register instead of a local register. This is a communication operation between the dual PEs used to ensure the data lines up with the coefficients.

Az = f0z + f7z
Bz = f1z + f6z
Cz = f2z + f5z
Dz = f3z + f4z
Ez = f3z - f4z
Fz = f2z - f5z
Gz = f1z - f6z
Hz = f0z - f7z

Next a sequence of eight multiply-add-send operations are completed followed by a scaling (shift) operation to conclude the 2-D DCT. Note that the rows are done in a different order than the columns were done. With the even rows in the first set of four 32 multiplication operations followed by the odd rows in the second set of four 32 multiplications. In the first 1-D DCT columns 0-3 were done first followed by columns 4-7. The end result is the same accomplishing the 2-D DCT on the whole 8x8 array. In the JPEG and MPEG algorithms a quantization step follows the 2-D DCT, in which case the scaling step can be included in the quantization step. The total number of cycles for an 8x8 2-D DCT (excluding scaling and quantization) is 18 cycles.

The problems and how they are solved summary:

1. provide a programmable low latency communication mechanism between processing elements in an array of processing elements,

The specification of the destination of results from functional execution units is changed from always the local processor's storage (register file), to any direct attached processor's storage (register file).

2. pipeline across the array of processing elements,

Due to the communication of results between processing elements with zero latency, computations can be pipelined across the array of processing elements.

3. communicate between clustered processing elements.

Specialized Data Select Unit instructions and expansion of the direct attached destination specification to include clustered processing elements provides the zero latency communications between the clustered processing elements and the ability to pipeline across the total interconnected array.

Although a specific embodiment of the invention has been disclosed, it will be understood by those having skill in the art that changes can be made to that specific embodiment without departing from the spirit and the scope of the invention.

Table 1

Nearest Neighbor Result Destination (For X substitute P/U/S)		
DEST	Single-PE Action	Dual-PE Action
0000	local GPRF Register ← X	local GPRF Register ← X
0001	local GPRF Register ← X	Transpose-PE GPRF Register ← X

THIS PAGE BLANK (USPTO)

DISCUSSION OF THE PREFERRED EMBODIMENT

Fig. 1 depicts a high level view of the Mwave array processor machine organization. The machine organization is partitioned into three main parts: the System Interfaces including Global Memory 100 and external I/O, multiple Control Units 103 with Local Memory 102, and the Execution Array with Distributed Control PEs 104. The System Interface is an application-dependent interface through which the Mwave array processor interfaces with Global Memory, the I/O, other system processors, and the personal computer/workstation host. Consequently, the System Interface will vary depending upon the application and the overall system design. The Control Units 103 contain the local memory 102 for instruction and data storage, instruction fetch (I-Fetch) mechanisms, and operand or data fetch mechanisms (D-Fetch). The Execution Array with Distributed Control PEs 104 is a computational topology of processing elements chosen for a particular application. For example, the array may consists of N Processing Elements (PEs) 104 per control unit 103, with each PE containing an Instruction Buffer (IMRY) 106, a General Purpose Register File (GPRF) 108, Functional Execution units (FNS) 110, Communication Facilities (COM) 112, and interfaces to its Instruction bus 114 and its data buses. The PEs may also contain PE-local instruction and data memories. Further, each PE contains an instruction decode register 116 shown in Figure 4, which supports distributed control of the multiple PEs. Synchronism of local memory accessing is a cooperative process between the control units 103, local memories 102, and the PEs 104. The array of PEs allows computation functions (FNS) to be executed in parallel in the PEs and results to be communicated (COM) between PEs.

With the multiple single instruction, multiple data (MSIMID) machine organization, e.g. Fig. 1, it is possible to create single or multiple thread machines wherein the topology of PEs and communication facilities can be configured for a more optimum topology depending upon the application. For example, some possible machine organizations are; multiple linear rings, a nearest neighbor 2-dimension mesh array, a folded nearest neighbor 2-dimension mesh, multiple-fold mesh, a 2-dimension hexagonal array, a folded 2-dimension hexagonal array, a folded mesh of trees, combinations of the above as well as others.

The basic concept involved in the Mwave array processor family is that the communications of results between direct-attached processing elements can be specified in the processing element instruction set architecture 115. In a typical RISC uni-processor, the destination of results from functional execution units is implied to be the processor's own register file breaks this "tradition", changing the definition of result destination, an implied single processor local destination, a directly-connected multiple processor destination. With this architecture and directly-connected PE-to-PE links, we can claim that communications between the directly connected PEs can be done with zero communication latency. The architecture provides this capability by including a destinatio field 120 in the simplex instructions 115 indicating the directly-connected succeeding PE where the result target register resides, which is specified by the target field 126. Figure 2 shows a basic simplex 32-bit instruction 115 used in the processor, though 16-bit, 64-bit, etc. formats can use the same principle. For a four PE neighborhood, a North, East, West, and South PE destination is coded in the Destination Extension field 120 is providing a 4-bit field here to allow for growth up to eight clustered processors (used in a three fold array) and up to an eight neighborhood array. Examples of topologies with direct-attached PEs are a nearest neighbor mesh, a folded mesh, a tree array, hypercubes, etc. It should also be appreciated that due to the clustering and nearest neighbor mesh organizations, the Physical Design process will physically place the directly connected PEs in close proximity to each other supporting short cycle times. Further, it should be appreciated that this binding of the communications destination specification in the instruction set format is done purposely to, not only provide 0-communication latency, but also to ensure hazard free communications in a SIMD array of processing elements. For explanation purposes only, the processor is used to describe the architectural details involved in implementing the described communications concept.

In the array processor instruction 115, the Operand-1 field 122, Operand-2 field 124, and Target field 126 are register specifications and the Destination field 120 specifies the direct attached succeeding processing element to which the result is to be communicated. The opcode field 128 specifies the arithmetic or logical operation to be performed by a selected execution unit.

In this type of SIMD array processors it is possible to tag the destination field as is shown in Figure 3, and use mode control instructions in the sequencer controller units to set the tag. The tags are then distributed to the PEs in the array. In order to change communications directions, a mode control instruction is issued prior to the execution of the instruction that communicates the result value. There are a number of implications in this approach. First, by tagging, the instruction has full use of the instruction field for function definition. Second, Tagging incurs an additional latency whenever the destination needs to be changed during communicating. If the loading of the tag register can be incorporated in a surrogate very large instruction word (VLIW), then it becomes possible to change the destination on a cycle by cycle basis. This minimizes the effect of this latency at the expense of complex control algorithms.

The single, diagonally folded array processor Element (PE) data flow will be briefly described. The folded-array nodes are of two types: diagonal nodes containing a single Processing Element, and all other nodes which contain two PEs each. The details of each type are discussed in the following sections.

Table 1 (continued)

Nearest Neighbor Result Destination (For X substitute P/U/S)		
DEST	Single-PE Action	Dual-PE Action
0010	Hyper-cube Complement-PE GPRF Register ← X	Hyper-cube Complement-PE GPRF Register ← X
0011	Reserved for Clustered Destinations	Reserved for Clustered Destinations
. . .	Reserved for Clustered Destinations	Reserved for Clustered Destinations
0111	Reserved for Clustered Destinations	Reserved for Clustered Destinations
1000 North	• W/N Out Port ← Local-X • TRGT Register ← S/E in Port South-X	• N/W Out Port ← Local-Xt • Top-PE TRGT Reg ← S/E in Port South-Xt • W/N Out Port ← Local-Xb • Bot-PE TRGT Reg ← E/S in Port South-Xb
1001 North East	Reserved (e.g. 8-NN: NE port)	Reserved (e.g. 8-NN: NE port)
1010 East	• S/E Out Port ← Local-X • TRGT Register ← W/N in Port West-X	• E/S Out Port ← Local-Xt • Top-PE TRGT Reg ← W/N in Port West-Xt • S/E Out Port ← Local-Xb • Bot-PE TRGT Reg ← N/W in Port West-Xb
1011 South East	Reserved (e.g. 8-NN: SE port)	Reserved (e.g. 8-NN: SE port)
1100 South	• S/E Out Port ← Local-X • TRGT Register ← W/N in Port North-X	• S/E Out Port ← Local-Xt • Top-PE TRGT Reg ← N/W in Port North-Xt • E/S Out Port ← Local-Xb • Bot-PE TRGT Reg ← W/N in Port North-Xb
1101 South West	Reserved (e.g. 8-NN: SE port)	Reserved (e.g. 8-NN: SE port)
1110 West	• W/N Out Port ← Local-X • TRGT Register ← S/E in Port East-X	• W/N Out Port ← Local-Xt • Top-PE TRGT Reg ← E/S in Port East-Xt • N/W Out Port ← Local-Xb • Bot-PE TRGT Reg ← S/E in Port East-Xb
1111 North West	Reserved (e.g. 8-NN: SE port)	Reserved (e.g. 8-NN: SE port)

Note: Single-PE nodes have two nearest-neighbor ports and Dual-PE nodes have four such ports. Use of these is depicted in Figure 6 on page 10 and Figure 7 on page 11 respectively. The notation Xt and Xb refers to the "top" and "bottom" PEs in a Dual-PE node shown in Figure 7 on page 11.

Table 2 (Page 1 of 3). Adjacent-PE Result Destination (For X substitute P/U/S)

DEST	Single-PE Action	Dual-PE Action
0000	local GPRF Register ← X	local GPRF Register ← X
0001	local GPRF Register ← X	Transpose-PE GPRF Register ← X
0010	Hyper-cube Complement-PE GPRF Register ← X	Hyper-cube Complement-PE GPRF Register ← X
0011	Reserved for Clustered Destinations	Reserved for Clustered Destinations
⋮	Reserved for Clustered Destinations	Reserved for Clustered Destinations
0111	Reserved for Clustered Destinations	Reserved for Clustered Destinations
1000 North	Even Row PEs • W/N Out Port ← Local-X • TRGT Register ← W/N In Port North-X Odd Row PEs • S/E Out Port ← Local-X • TRGT Register ← S/E In Port South-X	Even Row PEs • N/W Out Port ← Local-Xt • Top-PE TRGT Reg ← N/W In Port North-Xt • W/N Out Port ← Local-Xb • Bot-PE TRGT Reg ← W/N In Port North-Xb Odd Row PEs • S/E Out Port ← Local-Xt • Top-PE TRGT Reg ← S/E In Port South-Xt • E/S Out Port ← Local-Xb • Bot-PE TRGT Reg ← E/S In Port South-Xb
1001 North East	Reserved (e.g. 8-NN: NE port)	Reserved (e.g. 8-NN: NE port)

TABLE 2 (page 2 of 3)

Table 2 (Page 2 of 3). Adjacent-PE Result Destination (For X substitute P/U/S)		
DEST	**Single-PE Action**	**Dual-PE Action**
1010 East	Even Column PEs • S/E Out Port ← Local-X • TRGT Register ← S/E In Port East-X Odd Column PEs • W/N Out Port ← Local-X • TRGT Register ← W/N In Port West-X	Even Column PEs • E/S Out Port ← Local-Xt • Top-PE TRGT Reg ← E/S In Port East-Xt • S/E Out Port ← Local-Xb • Bot-PE TRGT Reg ← S/E In Port East-Xb Odd Column PEs • W/N Out Port ← Local-Xt • Top-PE TRGT Reg ← W/N In Port West-Xt • N/W Out Port ← Local-Xb • Bot-PE TRGT Reg ← N/W In Port West-Xb
1011 South East	Reserved (e.g. 8-NN: SE port)	Reserved (e.g. 8-NN: SE port)
1011 South	Even Row PEs • S/E Out Port ← Local-X • TRGT Register ← S/E In Port South-X Odd Row PEs • W/N Out Port ← Local-X • TRGT Register ← W/N In Port North-X	Even Row PEs • S/E Out Port ← Local-Xt • Top-PE TRGT Reg ← S/E In Port South-Xt • E/S Out Port ← Local-Xb • Bot-PE TRGT Reg ← E/S In Port South-Xb Odd Row PEs • N/W Out Port ← Local-Xt • Top-PE TRGT Reg ← N/W In Port North-Xt • W/N Out Port ← Local-Xb • Bot-PE TRGT Reg ← W/N In Port North-Xb
1101 South West	Reserved (e.g. 8-NN: SW port)	Reserved (e.g. 8-NN: SW port)

TABLE 2 (page 3 of 3)

Table 2 (Page 3 of 3). Adjacent-PE Result Destination (For X substitute P/U/S)		
DEST	**Single-PE Action**	**Dual-PE Action**
1001 West	Even Column PEs • W/N Out Port ← Local-X • TRGT Register ← W/N In Port West-X Odd Column PEs • S/E Out Port ← Local-X • TRGT Register ← S/E In Port East-X	Even Column PEs • W/N Out Port ← Local-Xt • Top-PE TRGT Reg ← W/N In Port West-Xt • N/W Out Port ← Local-Xb • Bot-PE TRGT Reg ← N/W In Port West-Xb Odd Column PEs • E/S Out Port ← Local-Xt • Top-PE TRGT Reg ← E/S In Port East-Xt • S/E Out Port ← Local-Xb • Bot-PE TRGT Reg ← S/E In Port East-Xb
1111 North West	Reserved (e.g. 8-NN: NW port)	Reserved (e.g. 8-NN: NW port)
Note: Single-PE nodes have two nearest-neighbor ports and Dual-PE nodes have four such ports. Use of these is depicted in Figure 6 on page 10 and Figure 7 on page 11 respectively. The notation Xt and Xb refers to the "top" and "bottom" PEs in a Dual-PE node shown in Figure 7 on page 11.		

TABLE 3

1. Butterfly calculation on each column Figure 14

- Az = Pz0 + Pz7
- Bz = Pz1 + Pz6
- Cz = Pz2 + Pz5
- Dz = Pz3 + Pz4
- Ez = Pz3 - Pz4
- Fz = Pz2 - Pz5
- Gz = Pz1 - Pz6
- Hz = Pz0 - Pz7

2. 1st 32 Multiplies column z=0-3, Send Column 0-3 Pairs of Butterfly Results South (Figure 15):

- $fz0\text{-}1 = Az(c4x)$
- $fz2\text{-}1 = Bz(c6x)$
- $fz4\text{-}1 = Cz(-c4x)$
- $fz6\text{-}1 = Dz(-c6x)$
- $fz1\text{-}1 = Ez(c7x)$

- $fz3\text{-}1 = Fz(-cx)$
- $fz5\text{-}1 = Gz(-cx)$
- $fz7\text{-}1 = Hz(c7x)$
- Send (Az,Hz), (Bz,Gz), (Cz,Fz), (Dz,Ez) to South PE

3. 2nd 32 Multiplies column z=0.3. Send Column 0-3 Pairs of Butterfly Results South (Figure 16):

- $fz0\text{-}2 = (fz0\text{-}1 = Az(c4x)) + Dz(c4x)$
- $fz2\text{-}2 = (fz2\text{-}1 = Bz(c6x)) + Az(c2x)$
- $fz4\text{-}2 = (fz4\text{-}1 = Cz(-c4x)) + Bz(-c4x)$
- $fz6\text{-}2 = (fz6\text{-}1 = Dz(-c6x)) + Cz(c2x)$
- $fz1\text{-}2 = (fz1\text{-}1 = Ez(c7x)) + Fz(c5x)$
- $fz3\text{-}2 = (fz3\text{-}1 = Fz(-cx)) + Gz(-c7x)$
- $fz5\text{-}2 = (fz5\text{-}1 = Gz(-cx)) + Hz(c5x)$
- $fz7\text{-}2 = (fz7\text{-}1 = Hz(c7x)) + Ez(-cx)$
- Send (Az,Hz), (Bz,Gz), (Cz,Fz), (Dz,Ez) to South PE

4. 3rd 32 Multiplies column z=0-3, Send Column 0-3 Paris of Butterfly Results South (Figure 17):

- $fz0\text{-}3 = (fz0\text{-}2 = Az(c4x) + Dz(c4x)) + Cz\text{-}(c4x)$
- $fz2\text{-}3 = (fz2\text{-}2 = Bz(c6x) + Az(c2x)) + Dz(-c2x)$
- $fz4\text{-}3 = (fz4\text{-}2 = Cz(-c4x) + Bz(-c4x)) + Az(c4x)$
- $fz6\text{-}3 = (fz6\text{-}2 = Dz(-c6x) + Cz(c2x)) + Bz(-c2x)$
- $fz1\text{-}3 = (fz1\text{-}2 = Ez(c7x) + Fz(c5x)) + Gz(c3x)$
- $fz3\text{-}3 = (fz=3\text{-}2 = Fz(-cx) + Gz(-c7x)) + Hz(c3x)$
- $fz5\text{-}3 = (fz5\text{-}2 = Gz(-cx) + Hz(c5x)) + Ez(c3x)$
- $fz7\text{-}3 - (fz7\text{-}2 = Hz(c7x) + Ez(-cx)) + Fz(c3x)$
- Send (Az,Hz), (Bz,Gz), (Cz,Fz), (Dz,Ez) to South PE

5. 4th 32 Multiplies column z=0-3, Send Column 0-3 Pairs or Butterfly Results South (Figure 18):

- $fz0\text{-}4 = (fz0\text{-}3 = Az(c4x) + Dz(c4x) + Cz(c4x)) + Bz(c4x)$
- $fz2\text{-}4 = (fz2\text{-}3 = Bz(c6x) + Az(c2x) + Dz(-c2x)) + Cz(-c6x)$
- $fz4\text{-}4 = (fz4\text{-}3 = Cz(-c4x) + Bz(-c4x) + Az(c4x)) + Dz(c4x)$
- $fz6\text{-}4 = (fz6\text{-}3 = Dz(-c6x) + Cz(c2x) + Bz(-c2x)) + Az(c6x)$
- $fz1\text{-}4 = (fz1\text{-}3 = Ez(c7x) + Fz(c5x) + Gz(c3x)) + Hz(cx)$
- $fz3\text{-}4 = (fz3\text{-}3 = Fz(-cx) + Gz(-c7x) + Hz(c3x)) + Ez(-c5x)$
- $fz5\text{-}4 = fz5\text{-}3 = Gz(-cx) + Hz(c5x) + Ez(c3x)) + Fz(c7x)$
- $fz7\text{-}4 = fz7\text{-}3 = Hz(c7x) + Ez(-cx) + Fz(c3x)) + Gz(-c5x)$
- Send (Az,Hz), (Bz,Gz), (Cz,Fz), (Dz,Ez) to South PE

Claims

1. A data processing system, comprising:

   a storage means for storing a plurality of instructions, each instruction including a first designation of a source register, a second designation of an execution unit operation, a third designation of an execution unit to output port routing, and a fourth designation of an input port to target register routing;

   a plurality of processing elements, each coupled by means of an instruction bus to said storage means, each of said processing elements receiving one of said instructions broadcast over said instruction bus;

   each of said processing elements comprising:

   an instruction register coupled to said instruction bus, for receiving said broadcast instruction;

   a register file coupled to said instruction register, said register file including a target register and a first operand

register which stores a first operand;

an execution unit coupled to said instruction register, said first instruction designation controlling selective coupling of said first operand register said execution unit to provide to it said first operand;

said second instruction designation controlling said execution unit to execute an operation on said first operand to produce a result operand;

at least a first and a second output ports having outputs respectively coupled to a first and a second succeeding ones of said processing elements;

a switching means coupled to said instruction register, said third instruction designation controlling said switching means to selectively couple said execution unit to either the first output port or the second output port, to provide said result operand to the first succeeding processing element or to the second succeeding processing element, respectively;

at least a first and a second input ports having inputs respectively coupled to a first and a second preceding ones of said processing elements, the first input port adapted to receive a first next operand from the first preceding processing element and the second input port adapted to receive a second next operand from the second preceding processing element; and

said fourth instruction designation controlling said switching means to selectively couple said target register to either said first one of said input ports, to provide said first next operand to said target register or said second one of said input ports, to provide said second next operand to said target register;

whereby single instruction, multiple data processing can be performed.

2. The data processing system of claim 1, wherein said first succeeding processing element further comprises:

a second instruction register coupled to said instruction bus, for receiving said broadcast instruction;

a second register file coupled to said instruction register, said second register file including a second target register;

at least two input ports, including a receiving one of which is coupled to said first output port, for receiving said result operand; and

a second switching means coupled to said second instruction register, said fourth instruction designation controlling said second switching means to selectively couple said second target register to said receiving input port, to provide said result operand to said second target register;

whereby single instruction, multiple data processing can be performed.

3. The data processing system of claim 1 or 2, which further comprises:

at least two execution units coupled to said instruction register, said first instruction designation controlling selective coupling of said first operand register said a first one of said at least two execution units to provide to it said first operand;

said second instruction designation controlling said first one of said execution units to execute an operation on said first operand to produce a first result operand; and

said third instruction designation controlling said switching means to selectively couple said first one of said execution units to either the first output port or the second output port, to provide said first result operand to the first succeeding processing element or to the second succeeding processing element, respectively.

4. The data processing system of anyone of claims 1 to 3 which further comprises :
an instruction sequencing means coupled to said storage means and to said instruction bus, for fetching said

instructions from said storage means and broad casting them to said plurality of processing elements.

5. A data processing method, comprising:

retrieving a plurality of instructions, each instruction including a first designation of a source register, a second designation of an execution unit operation, a third designation of an execution unit to output port routing, and a fourth designation of an input port to target register routing;

broadcasting one of said instructions to each of a plurality of processing elements;

controlling with said first instruction designation, a selective coupling of a first operand register with an execution unit in each of said processing elements, to provide a first operand;

controlling with said second instruction designation, said execution unit to execute an operation on said first operand to produce a result operand in each of said processing elements;

controlling with said third instruction designation, a switching means in each processing element to selectively couple said execution unit to either a first output port or a second output port, to provide said result operand to a first succeeding processing element or to a second succeeding processing element, respectively; and

controlling with said fourth instruction designation, said switching means to selectively couple a target register to either a first input port, to provide a first next operand to said target register or a second input port, to provide a second next operand to said target register;

whereby single instruction, multiple data processing can be performed.

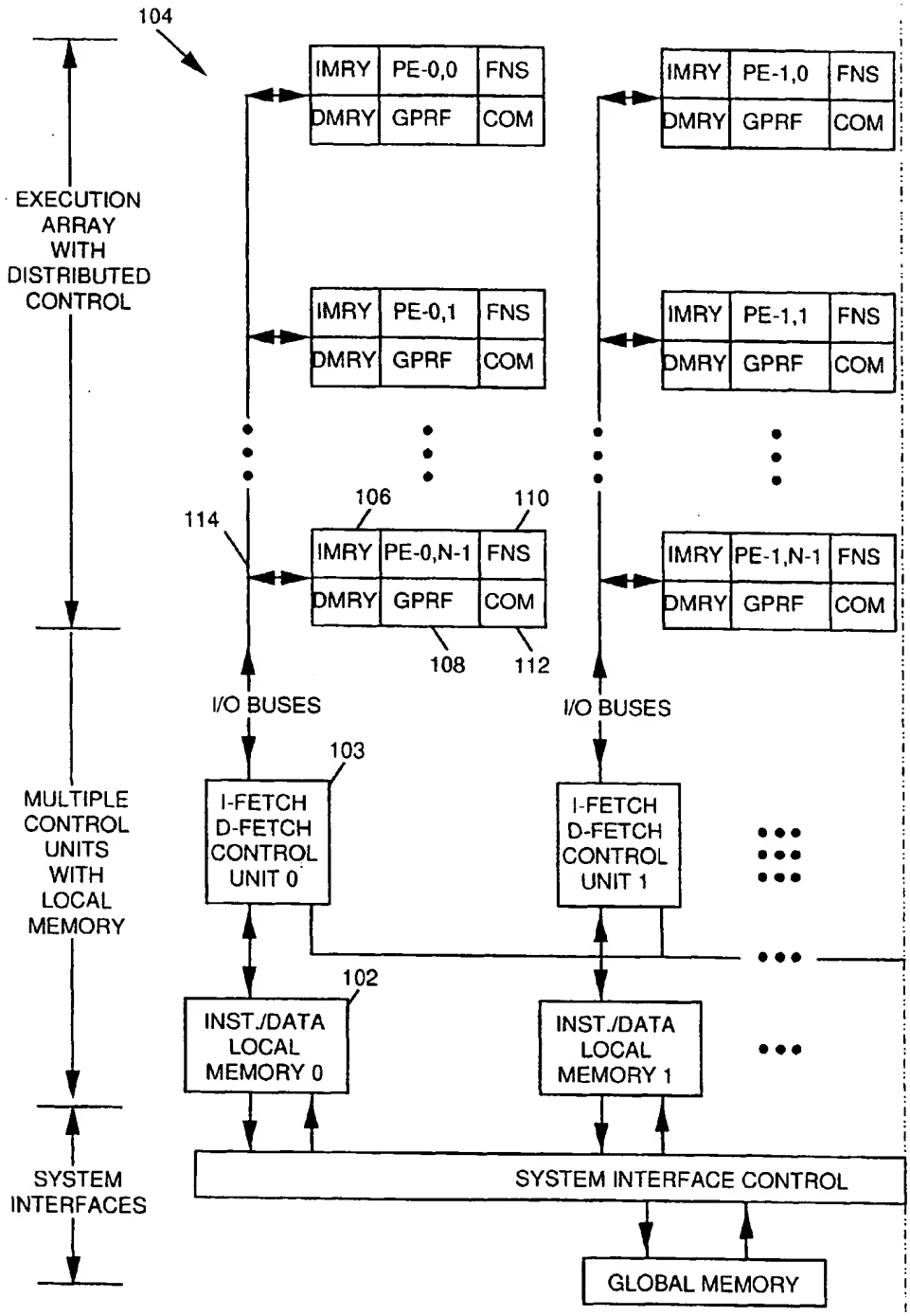
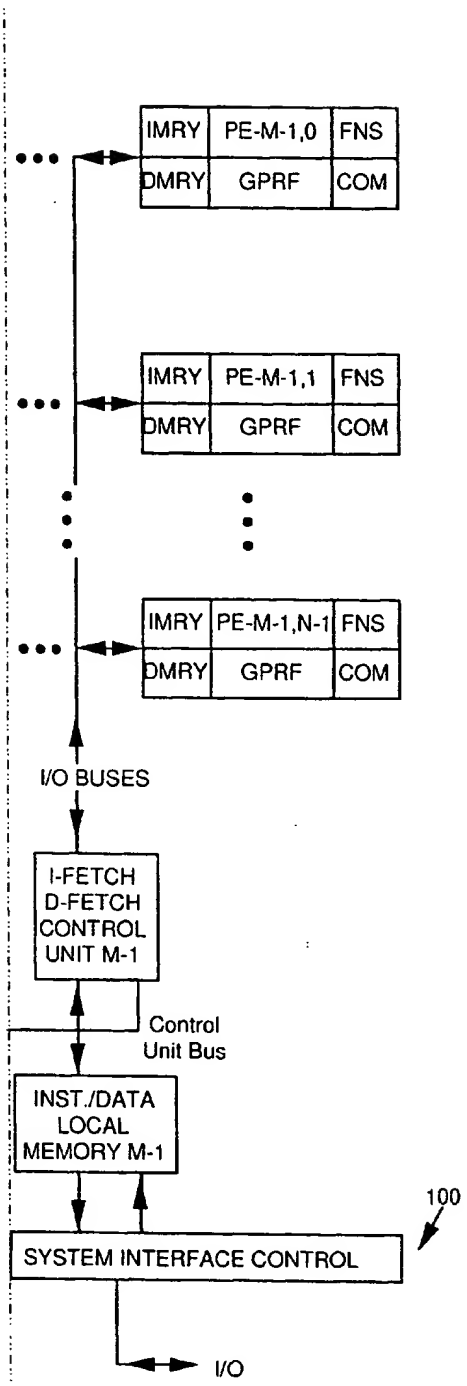

104
EXECUTION ARRAY WITH DISTRIBUTED CONTROL
IMRY
PE-0,0
FNS
DMRY
GPRF
COM
PE-1,0
PE-0,1
PE-1,1
106
110
114
PE-0,N-1
PE-1,N-1
108
112
I/O BUSES
103
MULTIPLE CONTROL UNITS WITH LOCAL MEMORY
I-FETCH D-FETCH CONTROL UNIT 0
I-FETCH D-FETCH CONTROL UNIT 1
102
INST./DATA LOCAL MEMORY 0
INST./DATA LOCAL MEMORY 1
SYSTEM INTERFACES
SYSTEM INTERFACE CONTROL
GLOBAL MEMORY

FIG 1A



High Level Array Machine Organization Multiple Control Units

FIG 1B

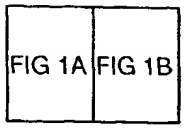
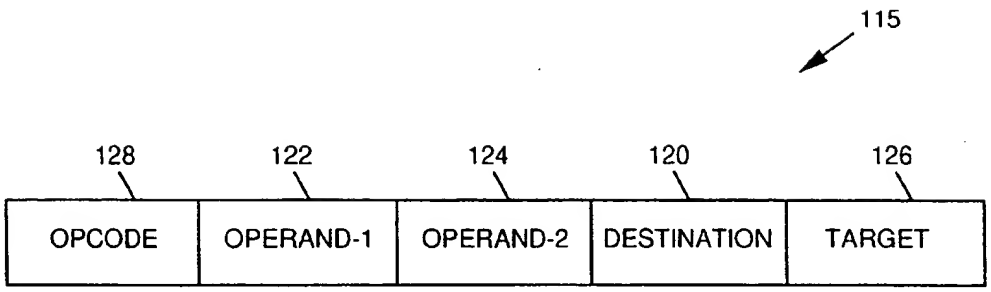
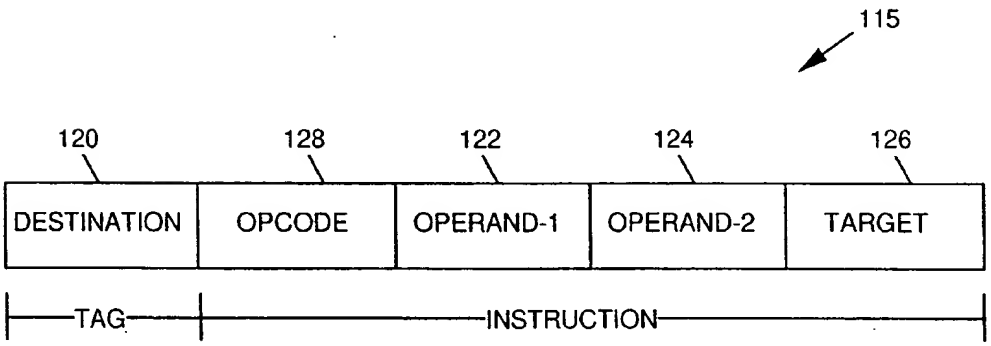


FIG 1



First Example Instruction Format for Communications

FIG 2



Second Example Instruction Format for Communications

FIG 3

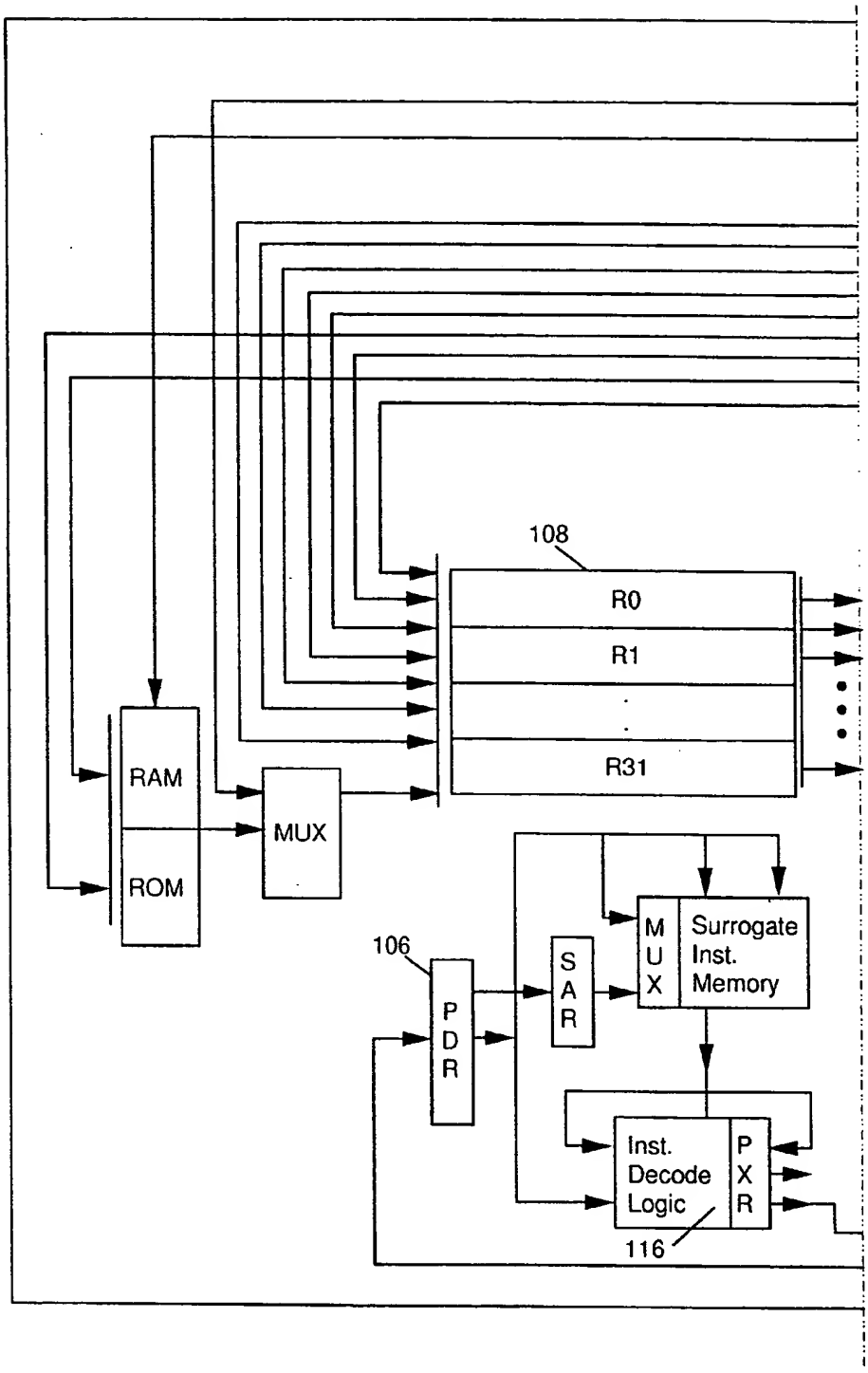
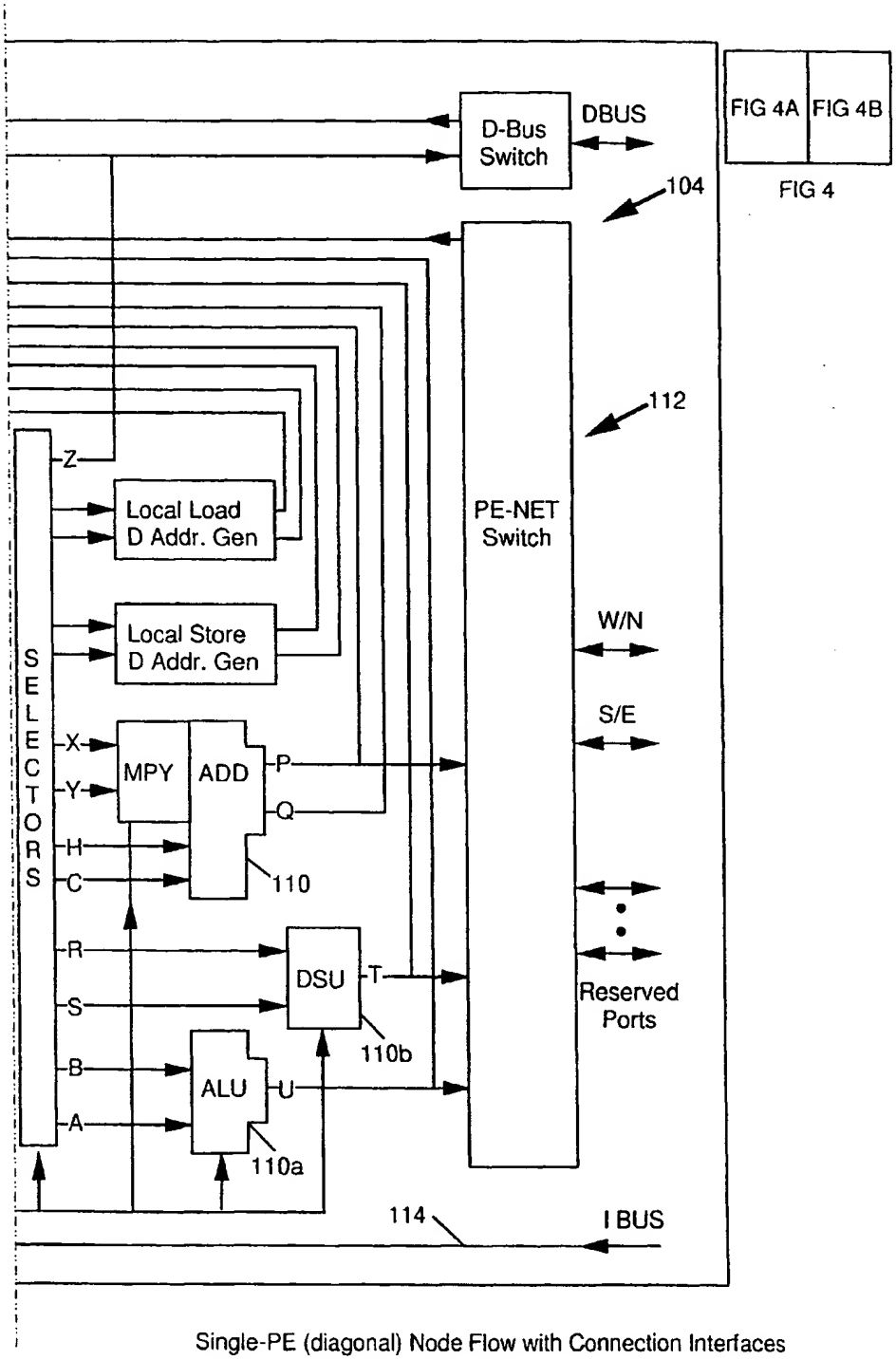

108
R0
R1
R31
RAM
ROM
MUX
106
PDR
SAR
MUX
Surrogate Inst. Memory
Inst. Decode Logic
PXR
116

FIG 4A



Single-PE (diagonal) Node Flow with Connection Interfaces

FIG 4B

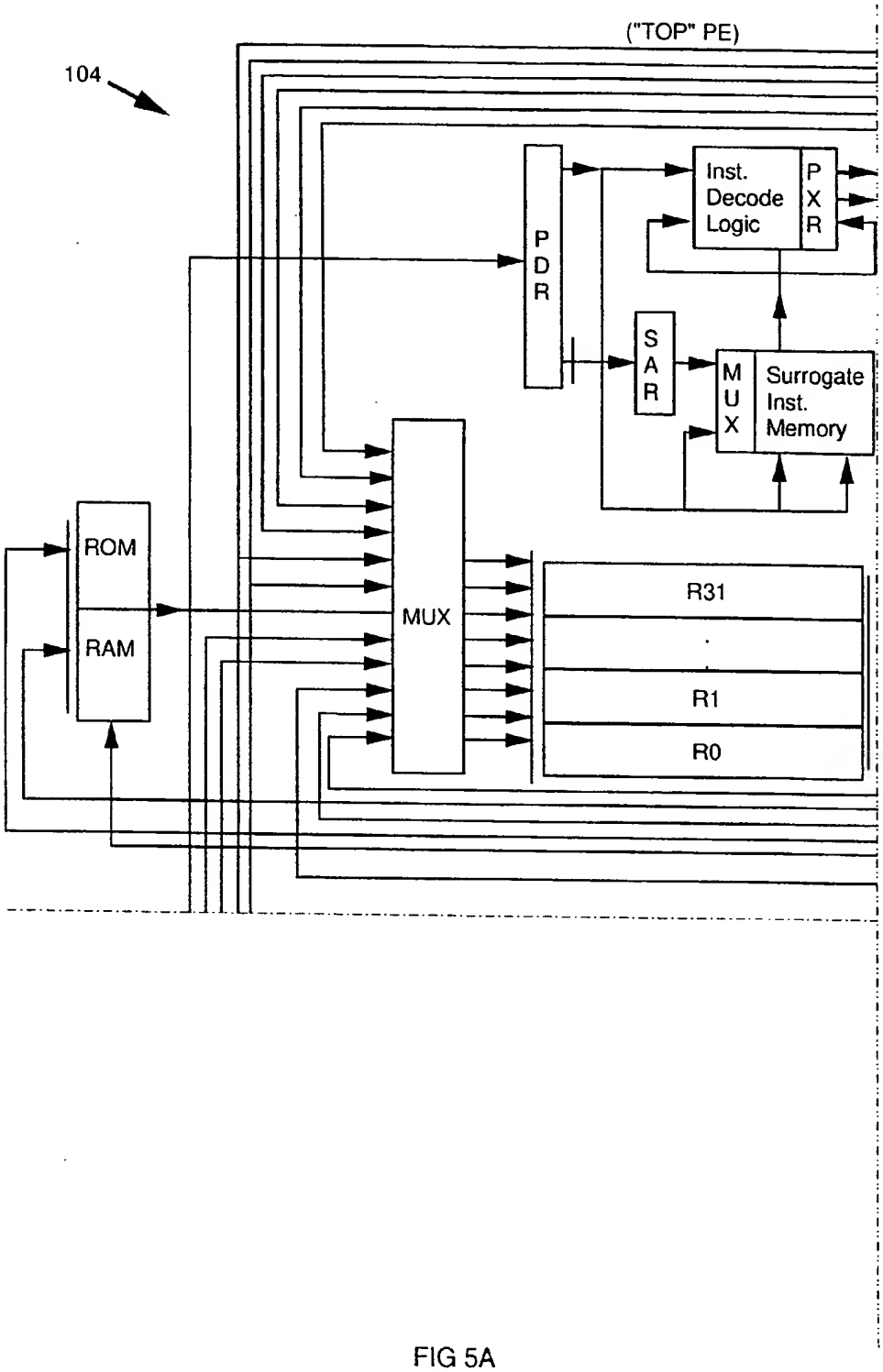

("TOP" PE)
104
Inst.
Decode
Logic
P
X
R
P
D
R
S
A
R
M
U
X
Surrogate
Inst.
Memory
ROM
RAM
MUX
R31
R1
R0

FIG 5A

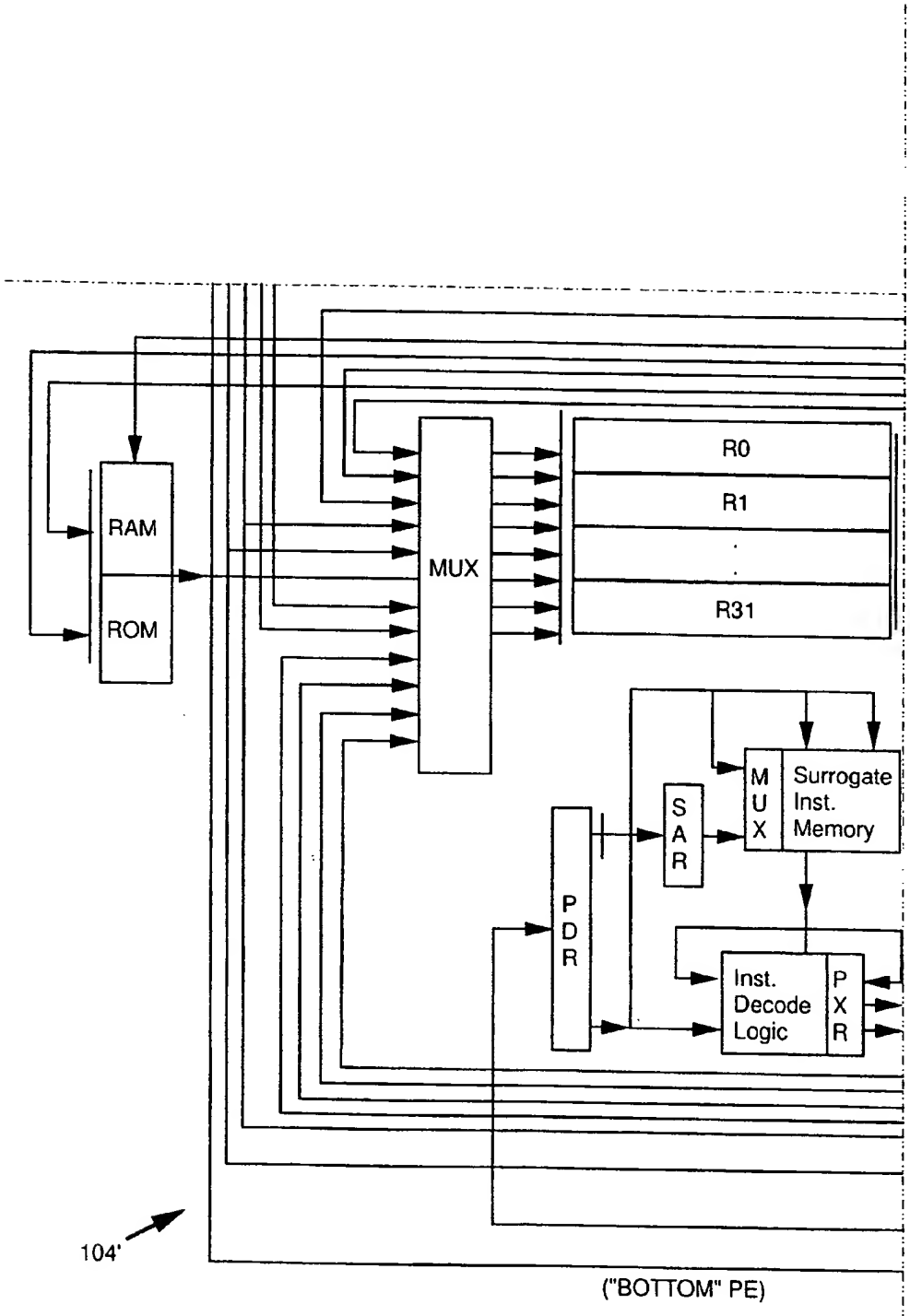

RAM
ROM
MUX
R0
R1
R31
M
U
X
Surrogate
Inst.
Memory
S
A
R
P
D
R
Inst.
Decode
Logic
P
X
R
104'
("BOTTOM" PE)

FIG 5B

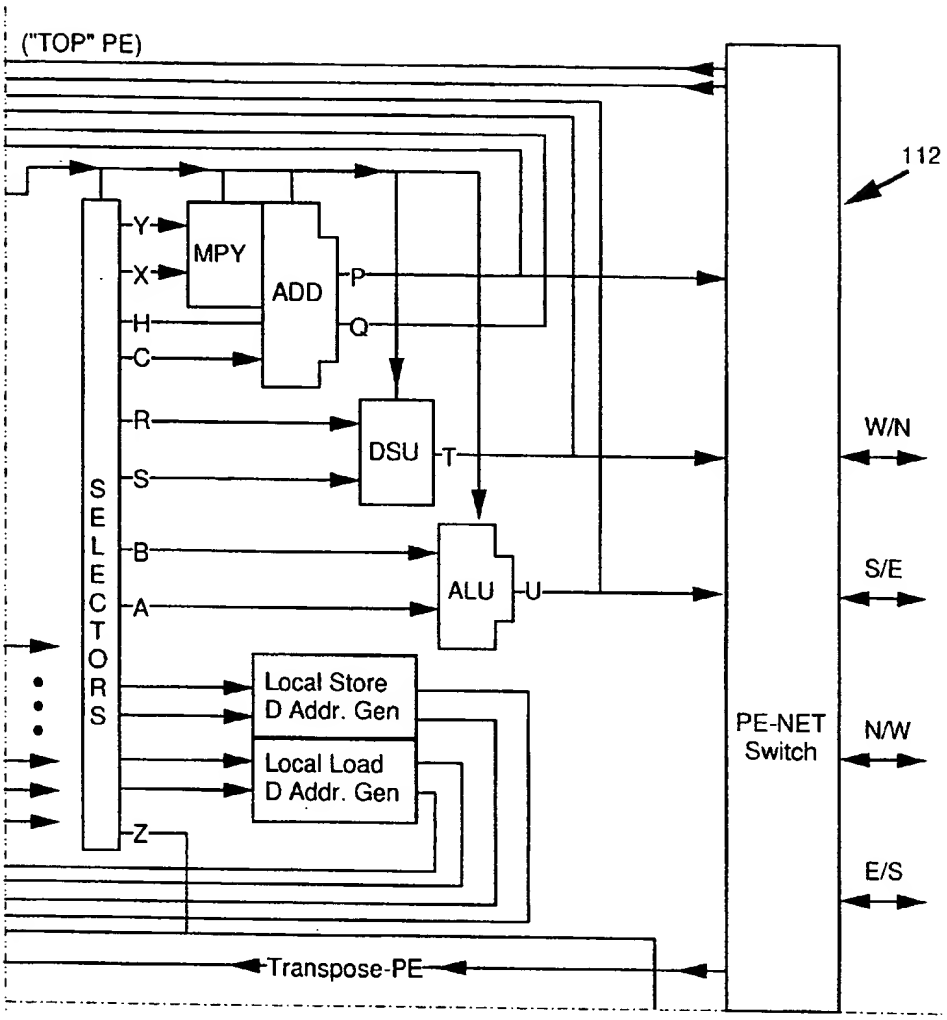

("TOP" PE)
112
Y
X
H
C
MPY
ADD
P
Q
R
S
DSU
T
B
A
ALU
U
SELECTORS
Local Store
D Addr. Gen
Local Load
D Addr. Gen
Z
PE-NET
Switch
W/N
S/E
N/W
E/S
Transpose-PE

FIG 5C

FIG 5A
FIG 5C
FIG 5B
FIG 5D

FIG 5

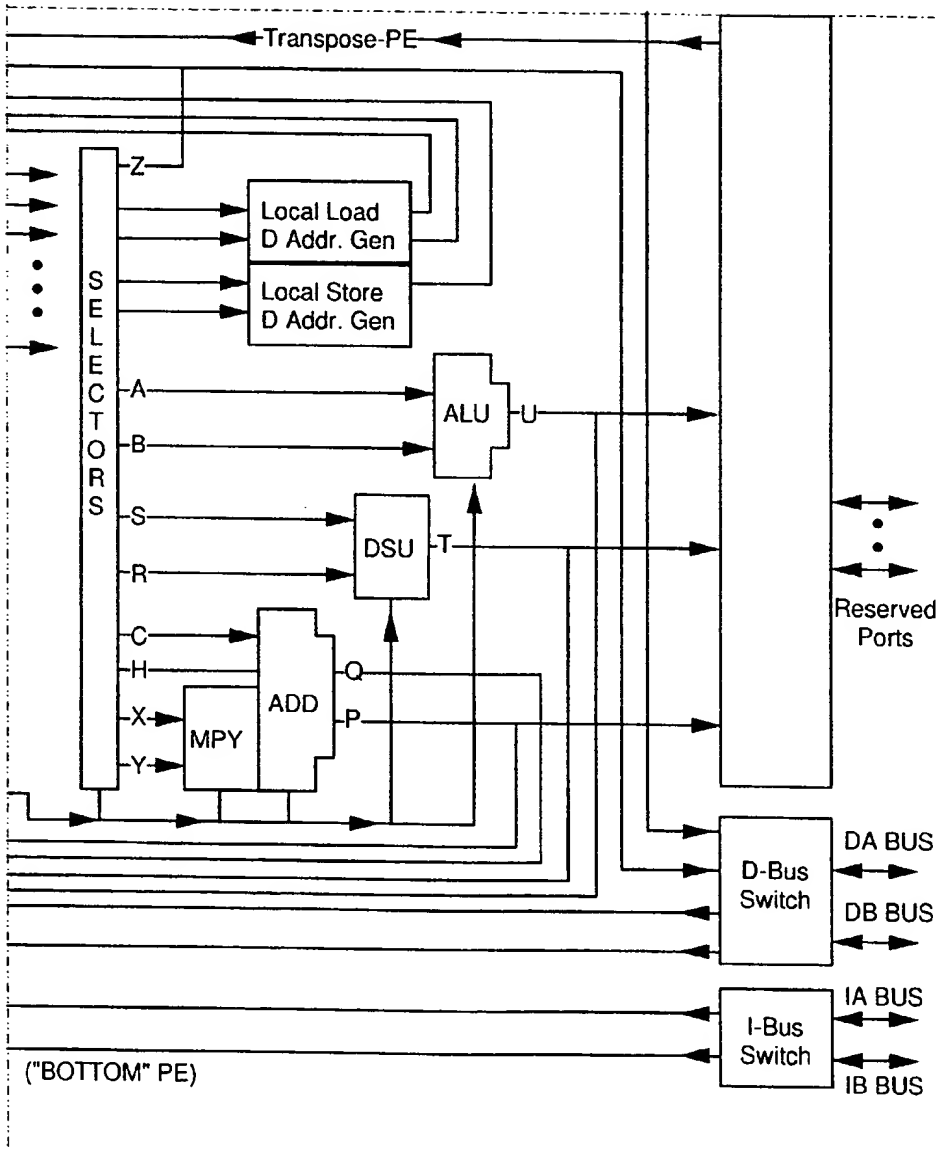
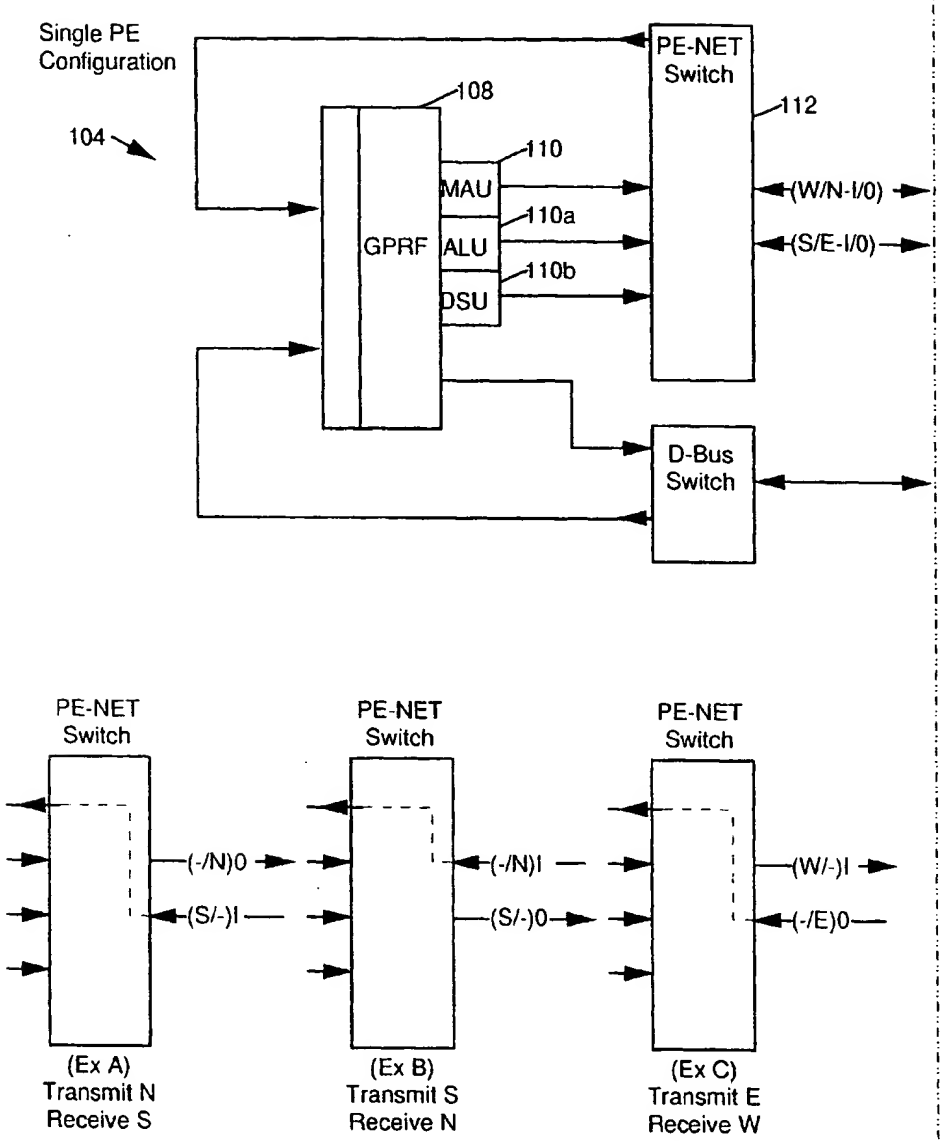

Transpose-PE
Z
SELECTORS
Local Load
D Addr. Gen
Local Store
D Addr. Gen
A
B
ALU
U
S
R
DSU
T
C
H
X
Y
MPY
ADD
Q
P
Reserved
Ports
D-Bus
Switch
DA BUS
DB BUS
I-Bus
Switch
IA BUS
IB BUS
("BOTTOM" PE)

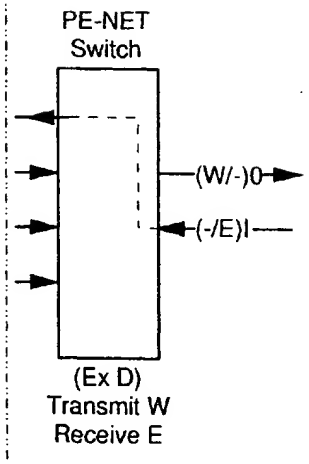
FIG 5D



(I, O represents the Input, Output, or bi-directional ports.)

Nearest-Neighbor Communication Examples in a Single-PE Node

FIG 6A



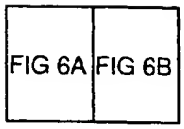
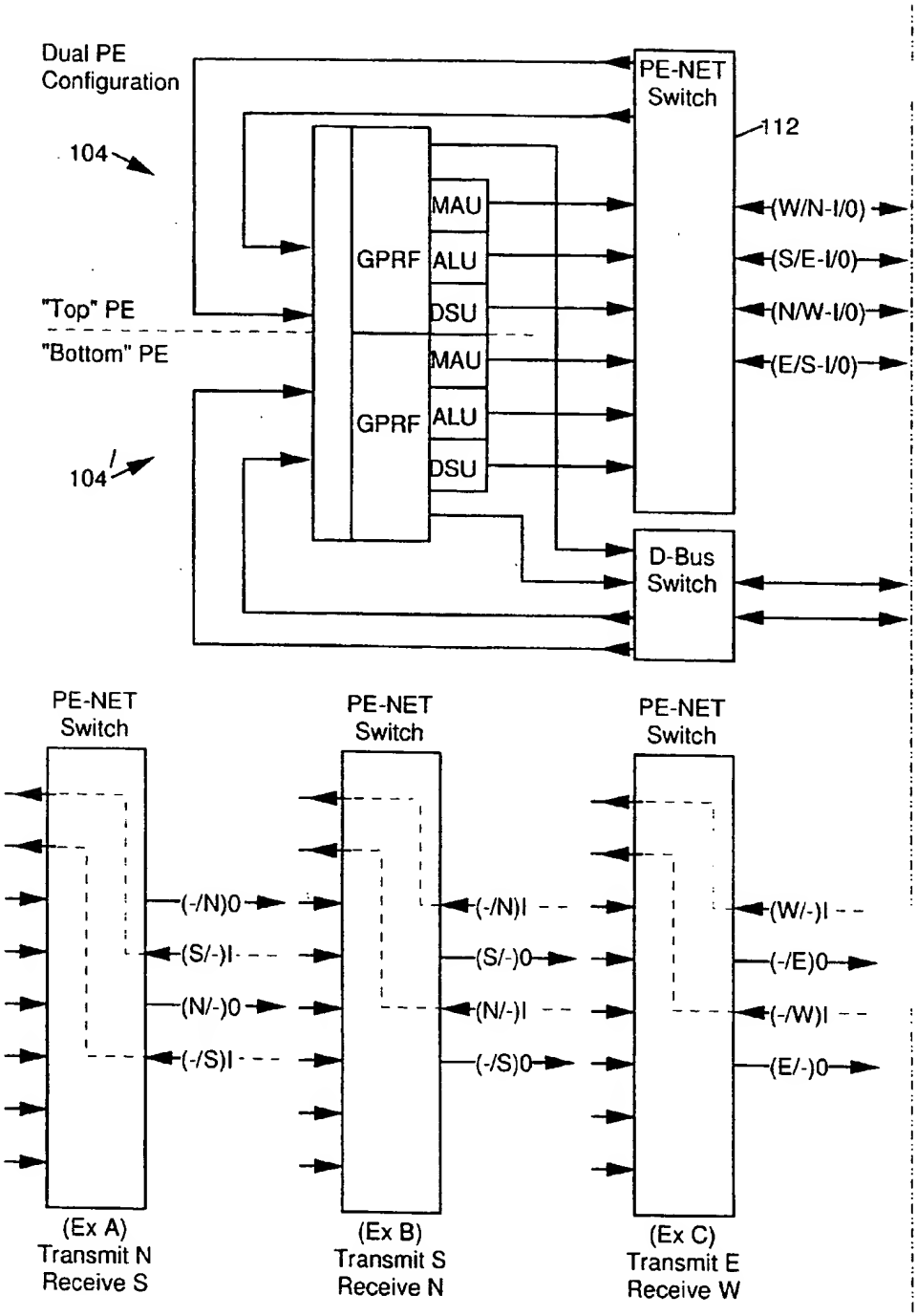


FIG 6

FIG 6B



(I, O represents the Input, Output or bi-directional ports.)

Nearest-Neighbor Communication Examples in a Dual-PE Node

FIG 7A

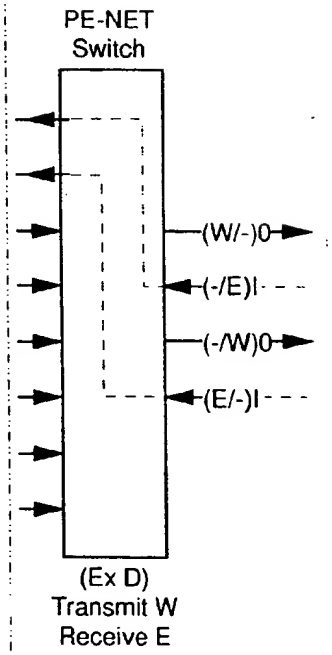


FIG 7B

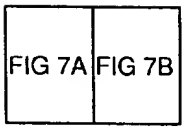


FIG 7

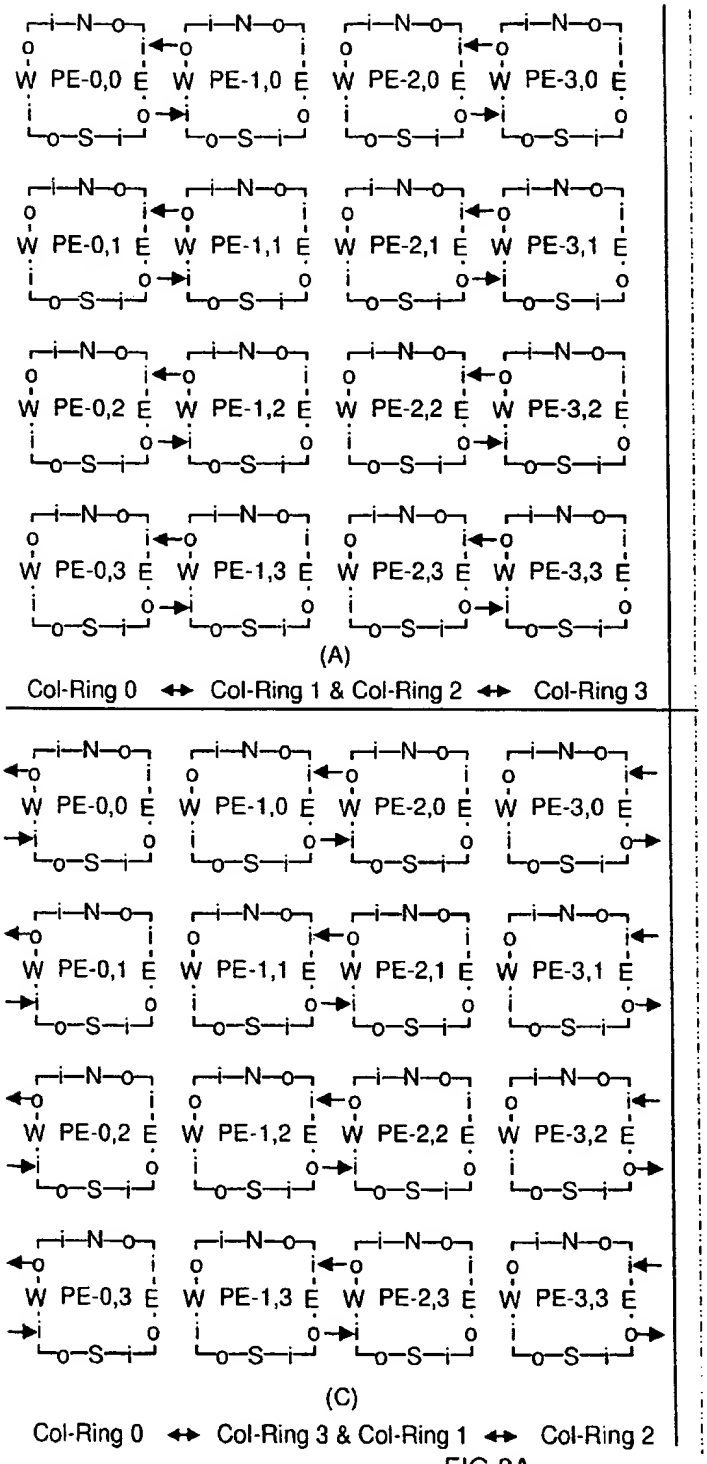

PE-0,0
PE-1,0
PE-2,0
PE-3,0
PE-0,1
PE-1,1
PE-2,1
PE-3,1
PE-0,2
PE-1,2
PE-2,2
PE-3,2
PE-0,3
PE-1,3
PE-2,3
PE-3,3
(A)
Col-Ring 0 ↔ Col-Ring 1 & Col-Ring 2 ↔ Col-Ring 3
(C)
Col-Ring 0 ↔ Col-Ring 3 & Col-Ring 1 ↔ Col-Ring 2

FIG 8A

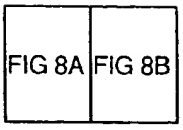
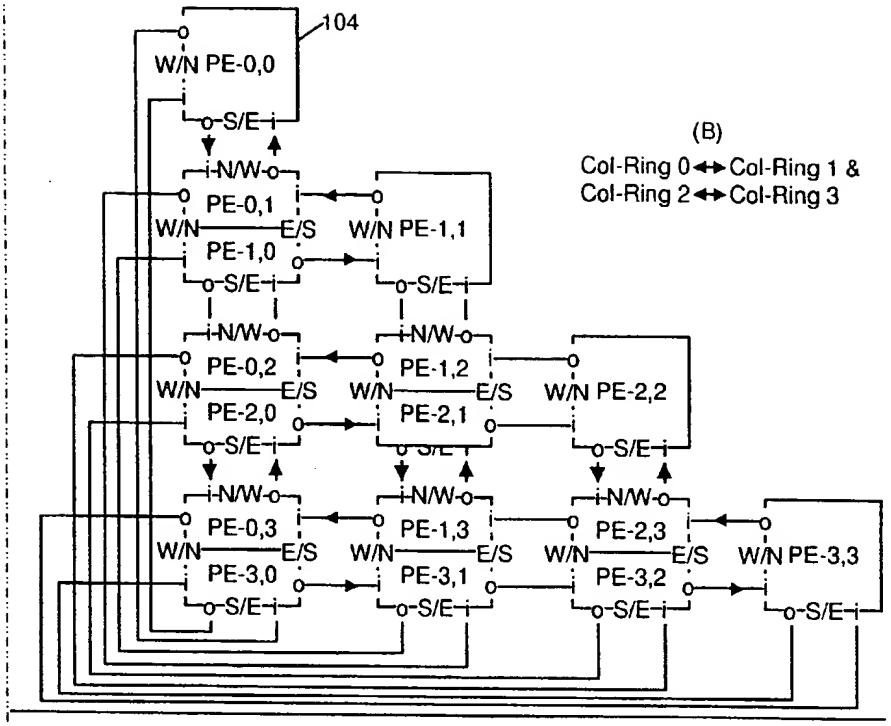


FIG 8A
FIG 8B

FIG 8


104
W/N PE-0,0
o-S/E-i
(B)
Col-Ring 0 ↔ Col-Ring 1 &
Col-Ring 2 ↔ Col-Ring 3
i-N/W-o
PE-0,1
W/N E/S
W/N PE-1,1
PE-1,0
PE-0,2
PE-1,2
W/N PE-2,2
PE-2,0
PE-2,1
PE-0,3
PE-1,3
PE-2,3
W/N PE-3,3
PE-3,0
PE-3,1
PE-3,2

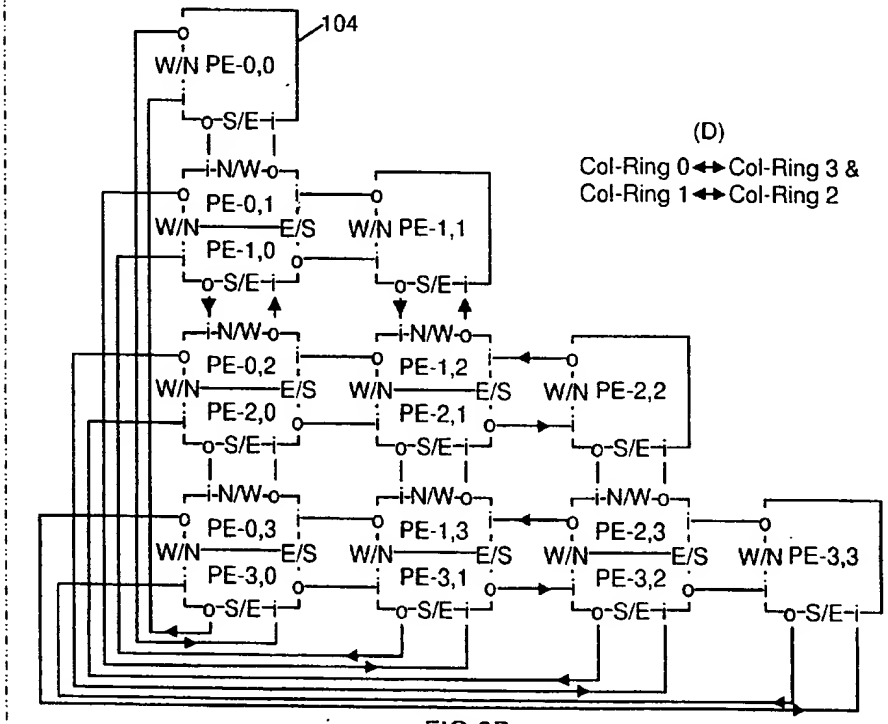

104
W/N PE-0,0
o-S/E-i
(D)
Col-Ring 0 ↔ Col-Ring 3 &
Col-Ring 1 ↔ Col-Ring 2
i-N/W-o
PE-0,1
W/N E/S
W/N PE-1,1
PE-1,0
PE-0,2
PE-1,2
W/N PE-2,2
PE-2,0
PE-2,1
PE-0,3
PE-1,3
PE-2,3
W/N PE-3,3
PE-3,0
PE-3,1
PE-3,2

FIG 8B

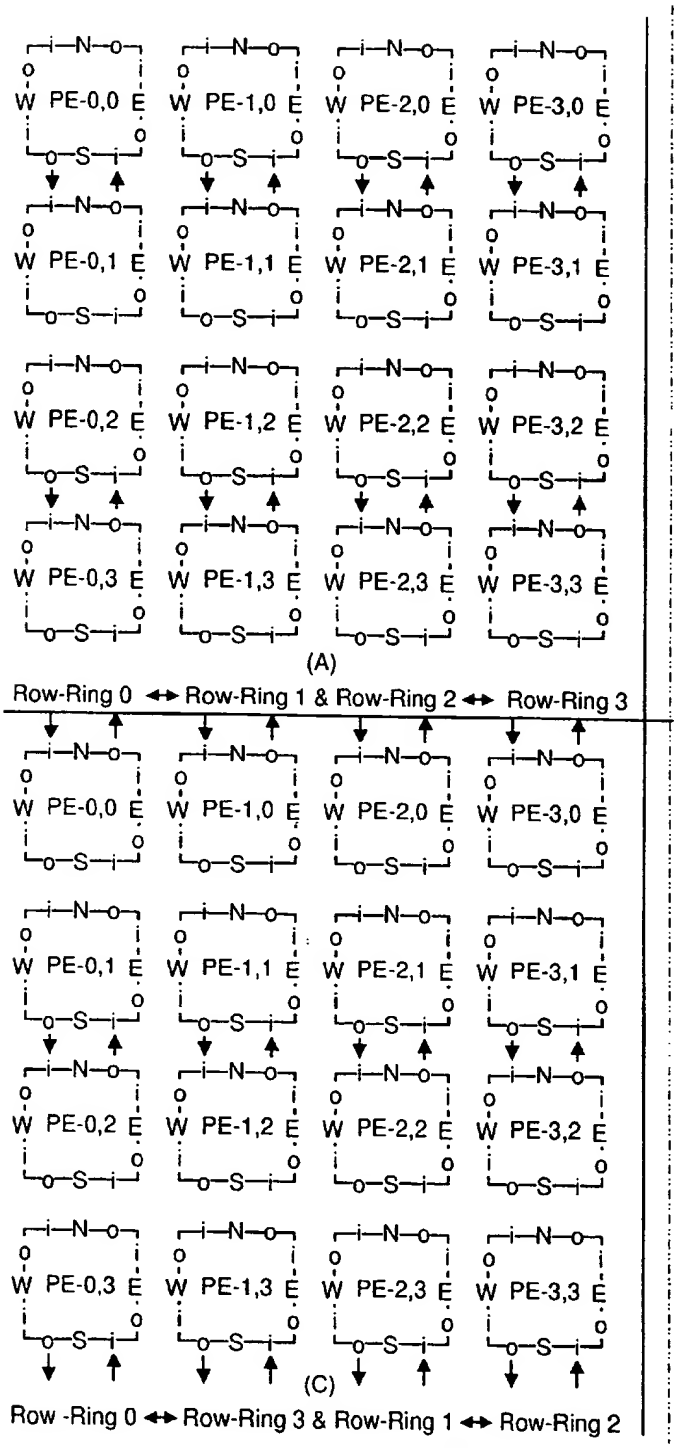

PE-0,0
PE-1,0
PE-2,0
PE-3,0
PE-0,1
PE-1,1
PE-2,1
PE-3,1
PE-0,2
PE-1,2
PE-2,2
PE-3,2
PE-0,3
PE-1,3
PE-2,3
PE-3,3
(A)
Row-Ring 0 ↔ Row-Ring 1 & Row-Ring 2 ↔ Row-Ring 3
(C)
Row -Ring 0 ↔ Row-Ring 3 & Row-Ring 1 ↔ Row-Ring 2

FIG 9A

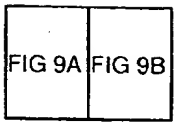


FIG 9A
FIG 9B

FIG 9

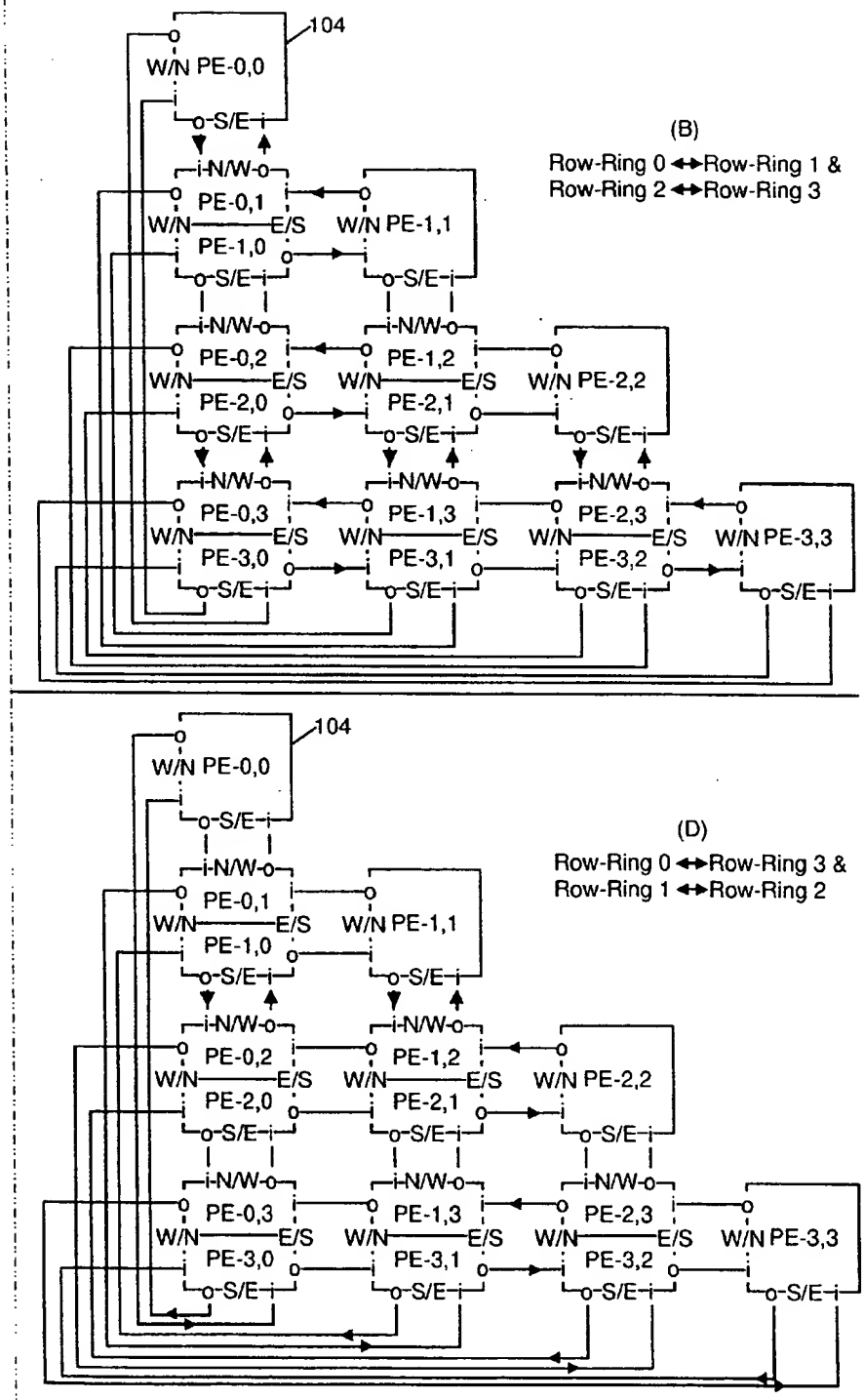
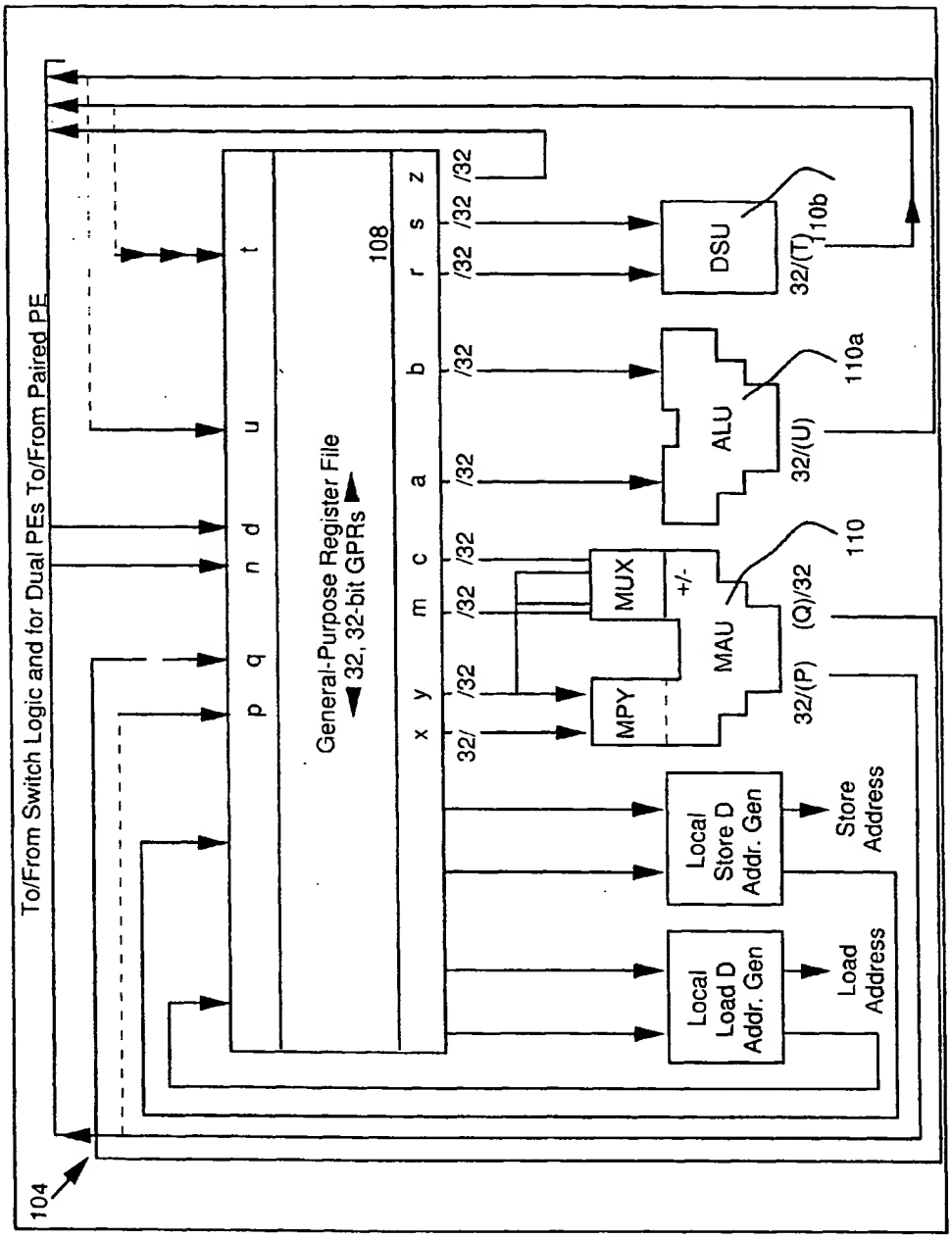
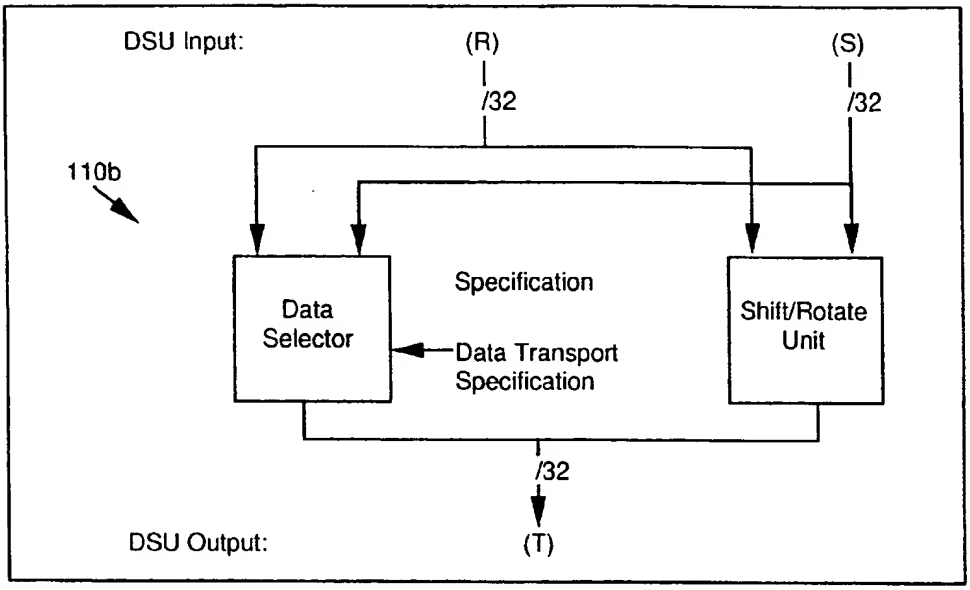

104
W/N PE-0,0
o-S/E-i
(B)
Row-Ring 0 ↔ Row-Ring 1 &
Row-Ring 2 ↔ Row-Ring 3
i-N/W-o
PE-0,1
W/N E/S
W/N PE-1,1
PE-1,0
PE-0,2
PE-1,2
W/N PE-2,2
PE-2,0
PE-2,1
PE-0,3
PE-1,3
PE-2,3
W/N PE-3,3
PE-3,0
PE-3,1
PE-3,2
104
W/N PE-0,0
(D)
Row-Ring 0 ↔ Row-Ring 3 &
Row-Ring 1 ↔ Row-Ring 2

FIG 9B



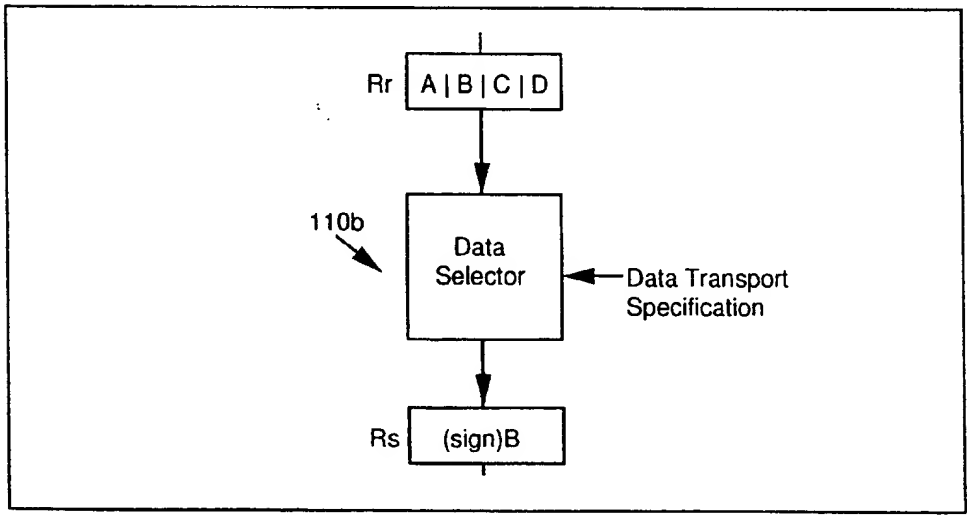
Flow Organization of a Processing Element.

FIG 10



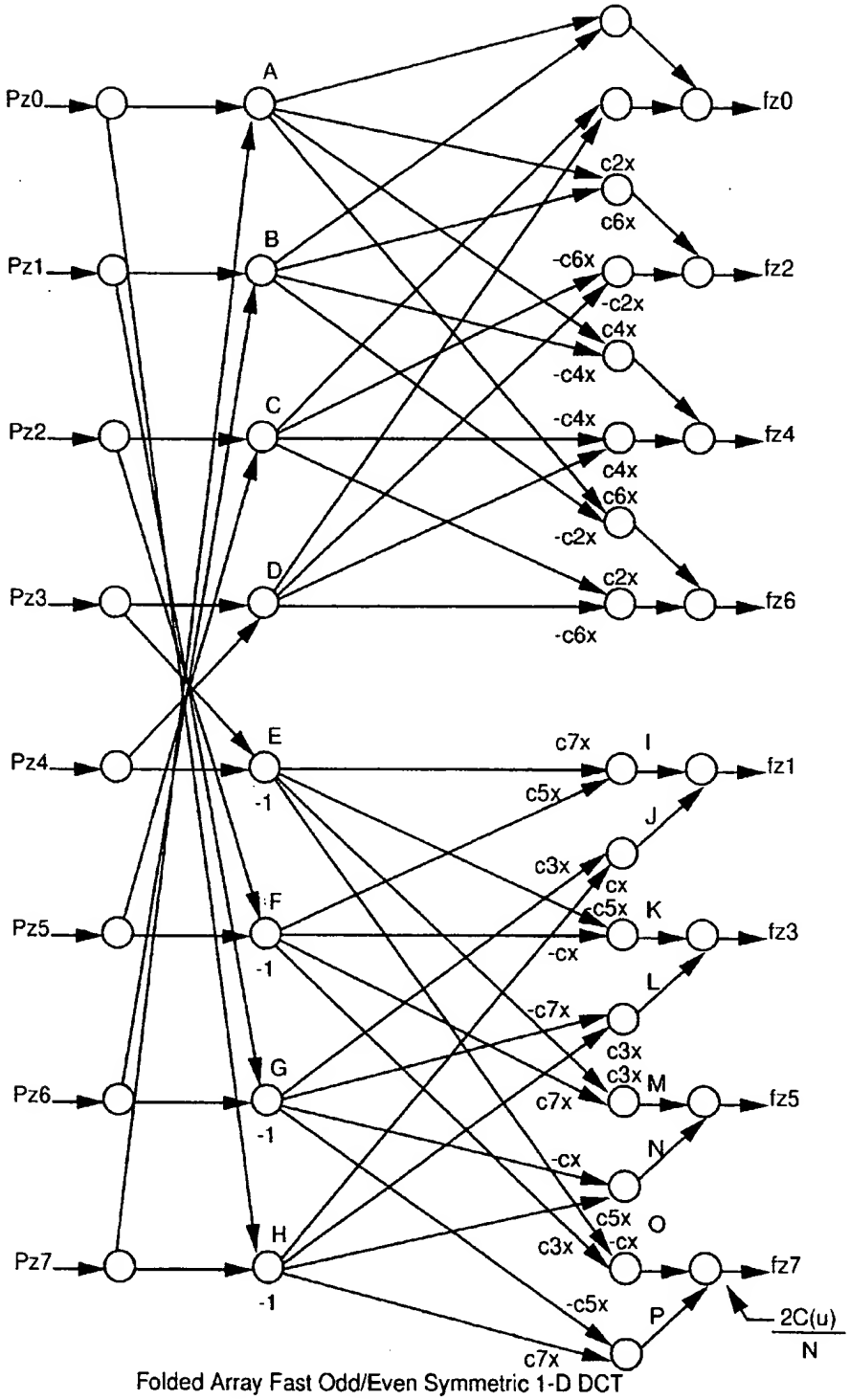
General Form of the Data Select Unit (DSU)

FIG 11



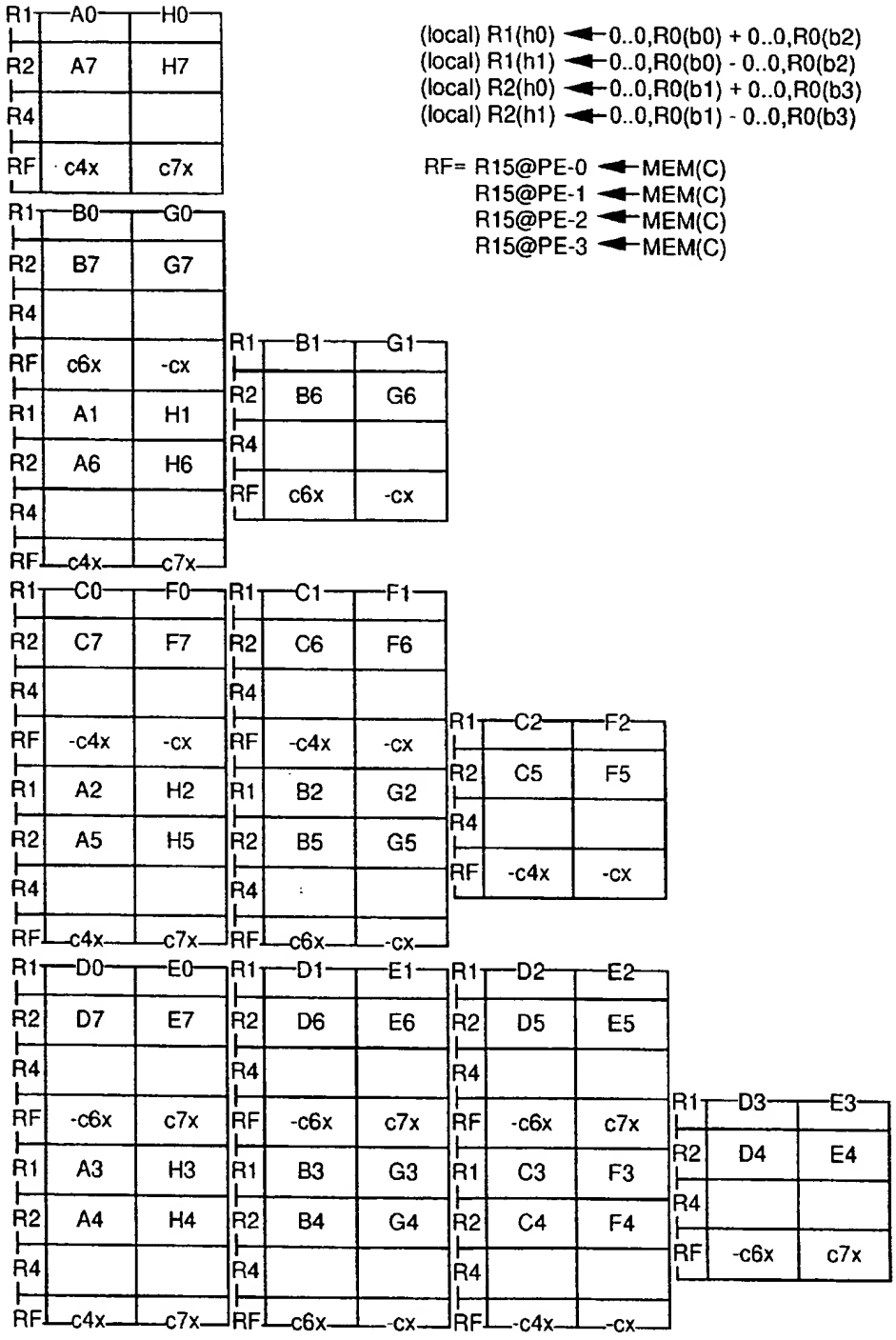
Example of DSU Use

FIG 12



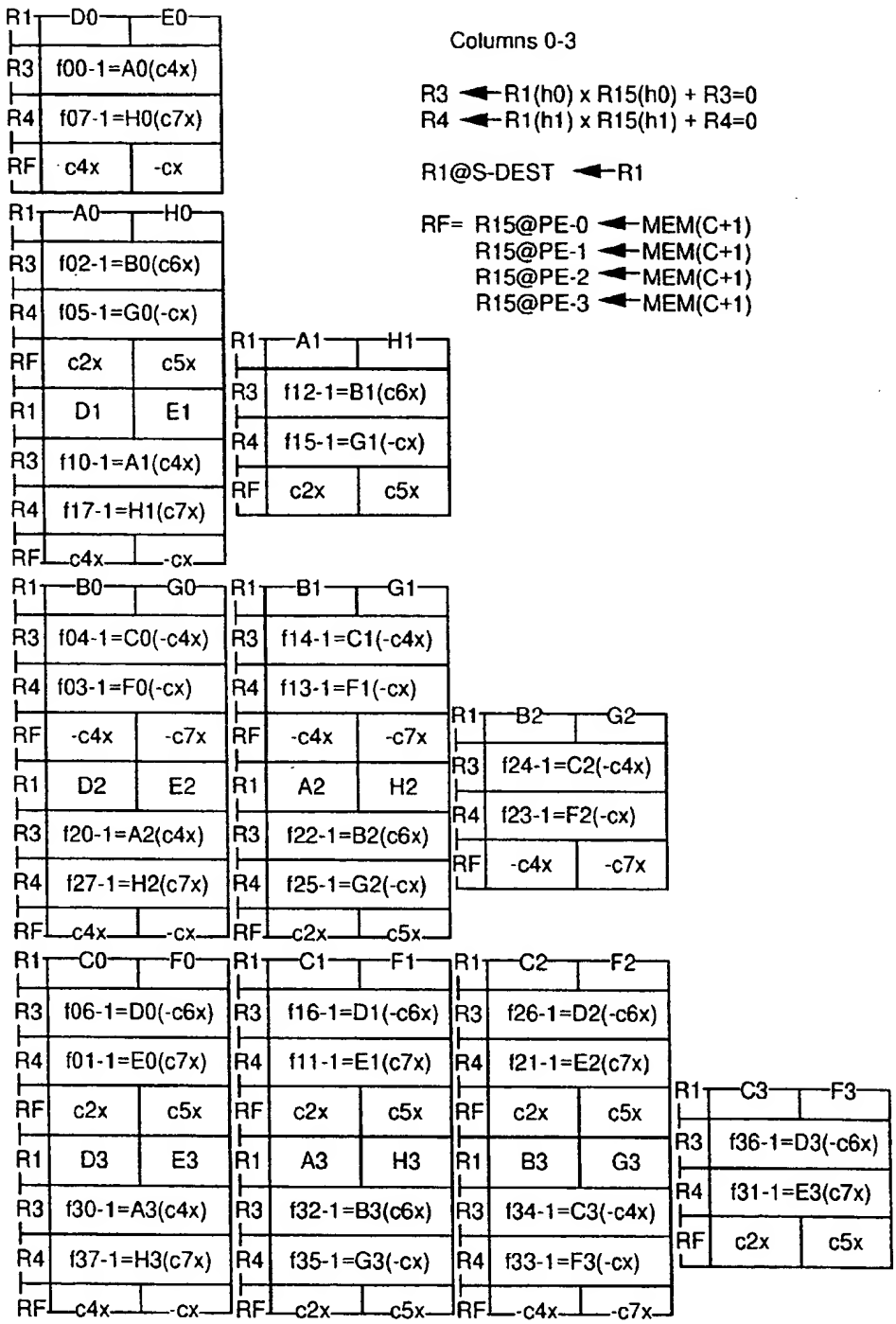
Folded Array Fast Odd/Even Symmetric 1-D DCT

FIG 13



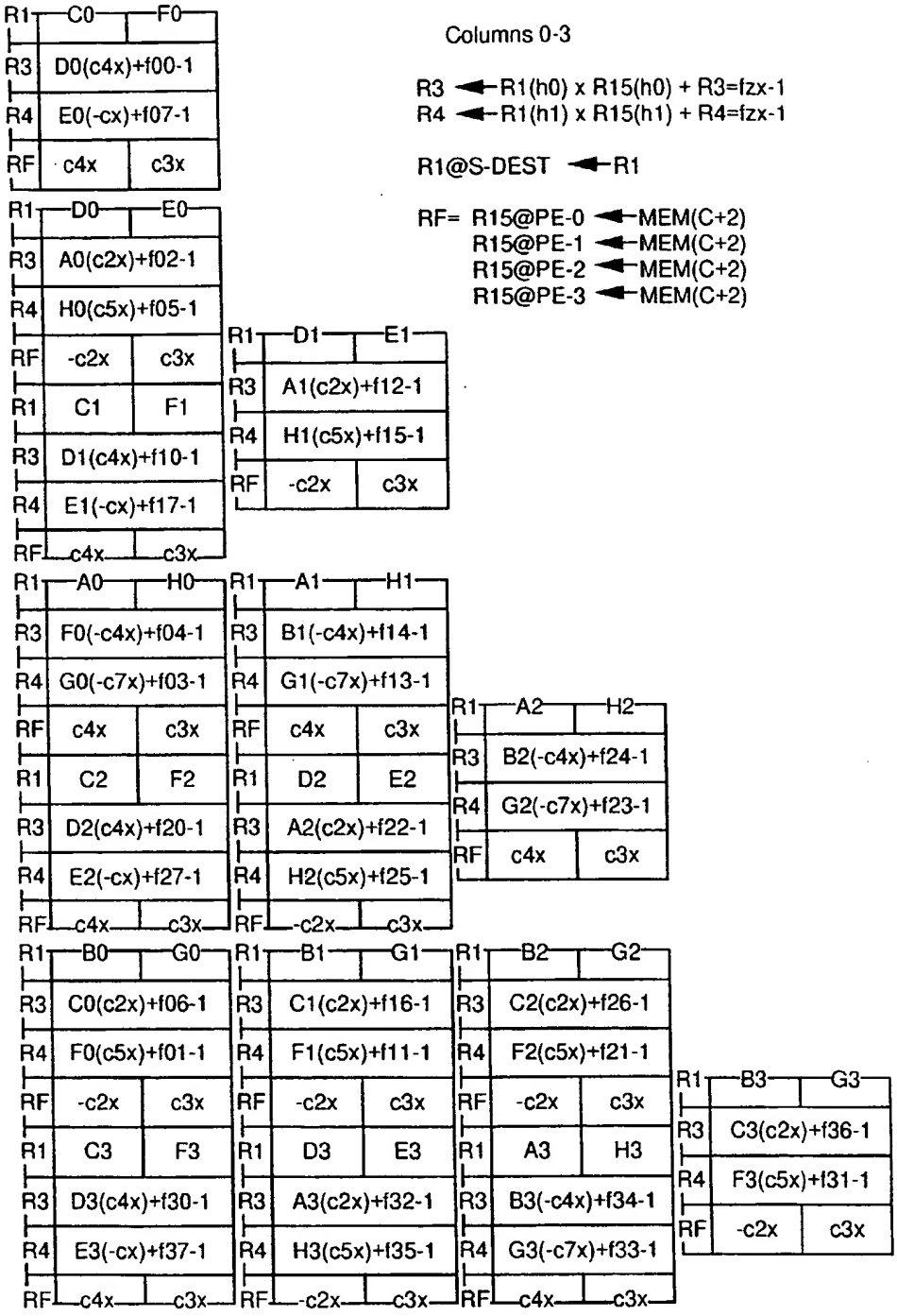
Execution of Butterfly Surrogate

FIG 14



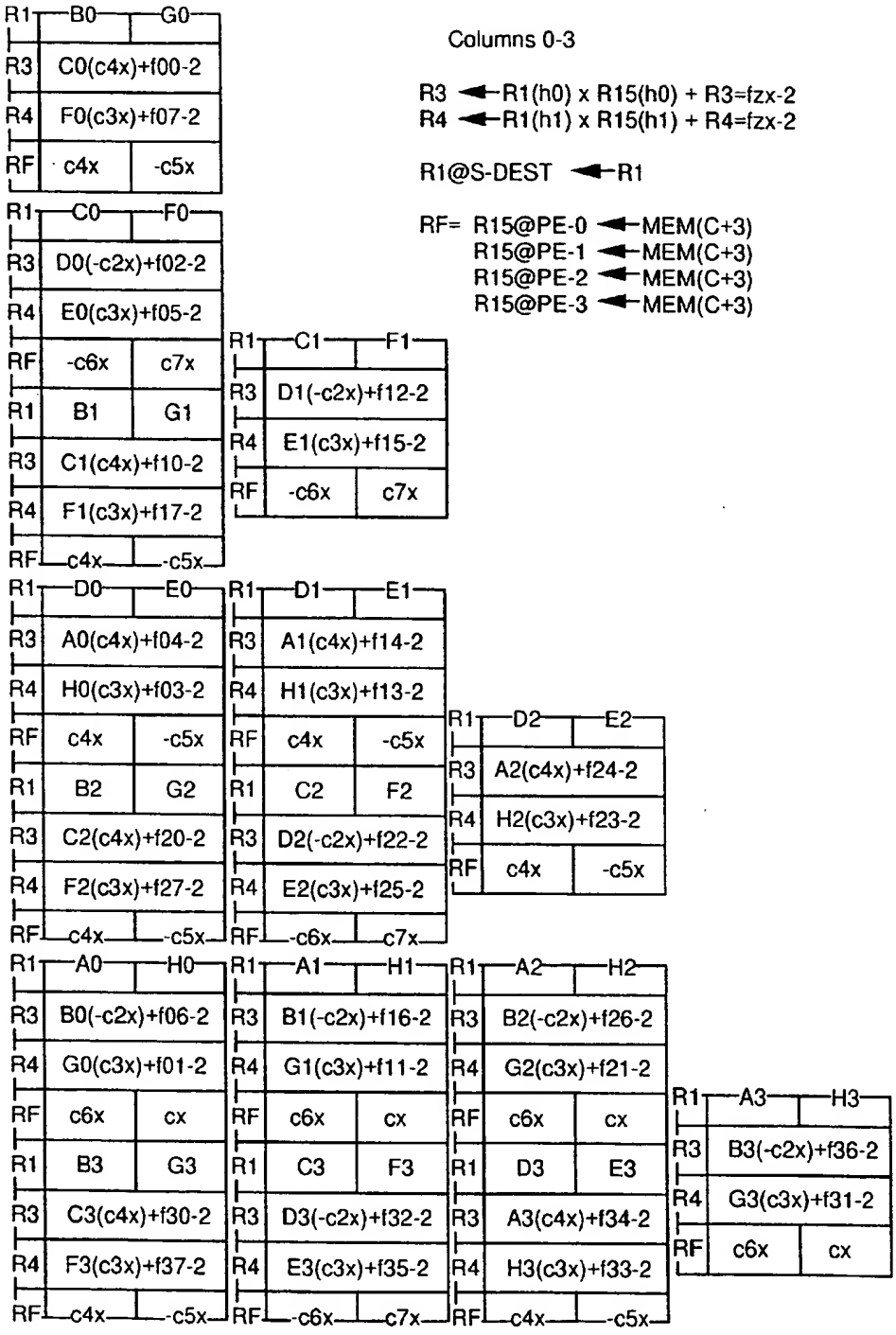
1st Execution of Multiply Add and DSU Send Surrogate

FIG 15



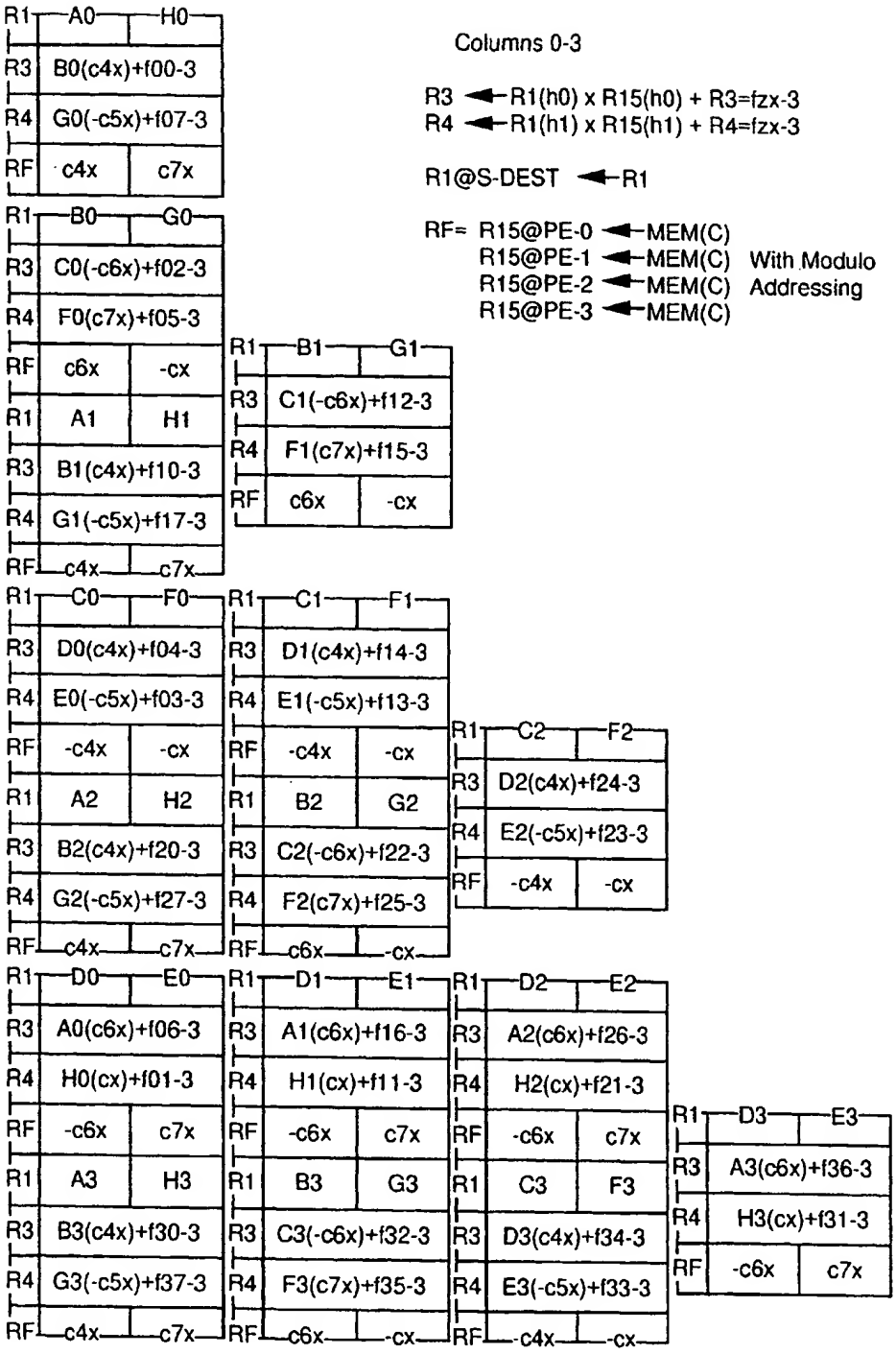
2nd Execution of Multiply Add and DSU Send Surrogate

FIG 16



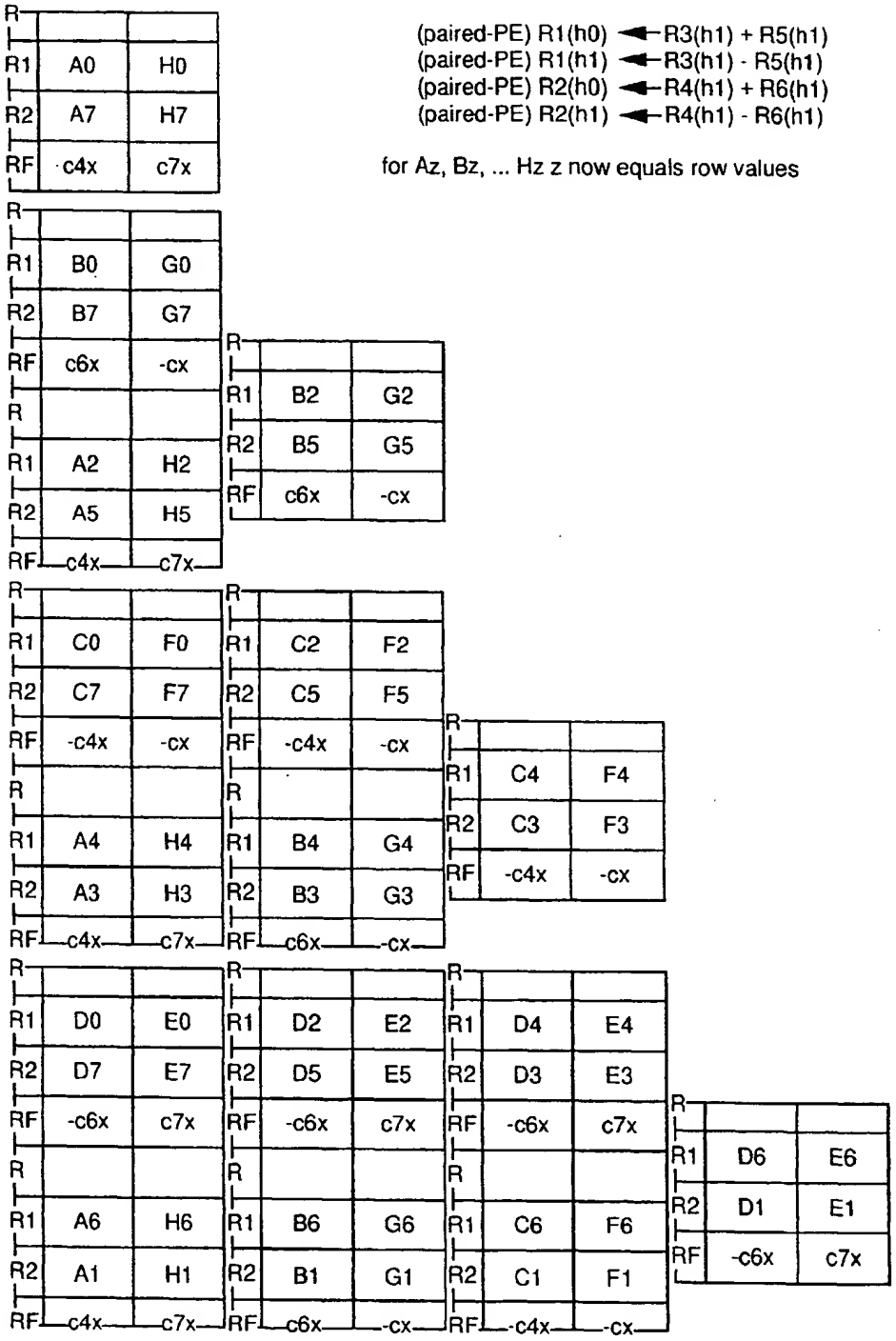
3rd Execution of Multiply Add and DSU Send Surrogate

FIG 17



4th Execution of Multiply Add and DSU Send Surrogate

FIG 18



Execution of Butterfly with Clustered PE Destination Surrogate

FIG 19